

# Soil Constitutive Modelling Using Neural Networks

Eleni Smyrniou

Master of Science Thesis



Civil Engineering and Geosciences  
Department of Geo-Engineering



# Soil Constitutive Modelling Using Neural Networks

---

*By*

Eleni Smyrniou

*In partial fulfilment of the requirements for the degree of*

*Master of Science*

*In Geo-engineering*

*at the Delft University of Technology,*

Thesis committee: Dr. P. J. Vardon, TU Delft  
Dr. ir. F. C. Vossepoel, TU Delft  
Dr.ir.T.Schweckendiek, TU Delft, Deltares

Company Supervisors: Dr. J. Nuttall, Deltares  
Dr. B. Zuada Coelho, Deltares



# PREFACE

---

This dissertation started in November 2017 in order to accomplish the requirements for the Geo-Engineering Master's degree at TU Delft. While working on the thesis I was able to acquire expertise on several fields and further develop my programming skills. I believe this process has prepared me adequately to become a junior engineer. Nevertheless, it would not be possible to deliver this thesis project on my own.

The thesis was conducted in collaboration with Deltares where I spend most of my time working on the project. Our team consisted of Jonathan Nuttall who assisted me on a day to day basis with his Neural Network knowledge. Furthermore, his critical thinking and general ability to point me to the right direction certainly kept me excited for the project and is one of the reasons I was able to look at the project from many different perspectives. The second member of the team was Bruno Zuada Coelho who assisted me in the Geotechnical Engineering part of the project. His attention to detail and general knowledge and experience in soil behaviour helped me resolve many issues I faced during the completion of the thesis project.

Also, I would like to thank the rest of my committee members P. Vardon, F. Vossepoel and T. Schweckendiek for their insightful guidance against any major problems encountered or decisions that had to be taken.

Moreover, I would like to thank my friends and family for their support who stood by me throughout my years as a student both in the Netherlands and in Greece.

*Eleni Smyrniou, 2018*

# Contents

---

PREFACE.....	iv
Contents.....	1
Abstract.....	3
Chapter 1. Introduction.....	5
1.1 Thesis Background.....	5
1.2 Proposed solution.....	5
1.3 Research Question.....	5
1.4 Scope of work .....	5
1.5 Report Outline.....	6
Chapter 2. Literature Study .....	7
2.1 Neural Networks- Basic Structure and Computational Operation .....	7
2.1.1 Structure-The Neuron .....	7
2.1.2 Structure-Connections and Full Neural Network.....	8
2.1.3 Computational Operation- Feed Forward Neural Network.....	8
2.1.4 Computational Operation- The back propagation method.....	10
2.1.5 Computational Operation- Activation functions .....	11
2.2 Applications of Neural Networks in Geotechnical engineering .....	12
2.3 Creating an Neural Network .....	14
2.3.2 Determination of the Neural Network’s architecture.....	15
2.3.3 Validation .....	17
2.3.4 Neural Network Callbacks.....	18
2.4 Different Types of Neural Networks.....	19
2.4.1 Nested Neural Networks.....	19
2.4.2 Feedback/Recurrent Neural Networks.....	20
2.5 Neural Networks for Constitutive models .....	21
2.5.1 Recurrent Neural Networks .....	21
2.5.2 Nested Neural Network .....	21
2.6 Neural Networks and Finite Element Analysis .....	22
2.6.1 “Intelligent finite element method” (Javadi & Tan, 2006).....	22
2.6.2 “A self-learning finite element code” (Shin & Pande, 2000).....	22
2.7 Conclusion of Literature Review .....	23
Chapter 3. First Approach- Generic Neural Network.....	24
3.1 Introduction.....	24
3.2 Methodology .....	24
3.2.1 Creating the dataset.....	24
3.2.2 Training Dataset.....	25
3.2.3 Creating the Generic Neural Network .....	27
3.3 Results.....	28

3.3.1	Network Architecture .....	28
3.3.2	Results of the Dataset .....	30
3.3.3	Improve Neural Network with Feedback Prediction .....	35
3.4	Conclusions .....	39
Chapter 4.	Second Approach-Component Based Modelling .....	40
4.1	Introduction .....	40
4.2	Methodology .....	40
4.2.1	Typical components of material models .....	40
4.2.2	Create the Training Datasets .....	42
4.2.3	Creation of Neural Networks .....	47
4.3	Results Linear- Elastic Model .....	49
4.3.1	Neural Network Model with One Direction as Input .....	49
4.3.2	Neural Network Model with Three Directions as Input .....	56
4.4	Linear- Elastic Perfectly Plastic Model .....	59
4.4.1	Defining Network Architecture and Training Results .....	59
4.4.2	Validation of Neural Network .....	63
4.5	Linear-Elastic Work hardening Model .....	72
4.5.1	Defining Network Architecture and Training Results .....	72
4.6	Conclusions of Component Based Neural Network Models .....	74
4.6.1	Linear- Elastic Neural Network .....	74
4.6.2	Linear-Elastic Perfectly-Plastic .....	75
4.6.3	Linear-Elastic Work-Hardening Model .....	76
Chapter 5.	Discussion .....	77
Chapter 6.	Conclusions .....	80
Chapter 7.	Recommendations for Future Research .....	82
Appendix A	.....	84
Appendix B	.....	86
Appendix C	.....	87
Bibliography	.....	89

# Abstract

---

Constitutive models are one of the main building blocks of the Finite Element Analysis that nowadays is used in almost every geotechnical engineering project. Thus, finding realistic stress-strain behaviour models has been one of the main fields of research in Geotechnical Engineering. However, constitutive equations have become increasingly complex featuring 10 or more parameters as inputs with sometimes small correlations to physical properties (Beaty & Byrne, 1998; Bauer, 1996). Thus, a more data driven approach can be determined to account for that issue. In this thesis, that data driven approach will be attempted by using Neural Networks. The main goal of the thesis is to assess if Neural Networks can be used to model constitutive soil behaviour. Specifically, two approaches are used to model stress-strain behaviour of soils.

The first approach is classified as a generic approach because the investigation is mostly focused on which techniques of the Neural Network can help with the modelling of stress and strain behaviour. In that way the Network can be thought of as a “black box”. The prediction after the training of the Neural Network is achieved by dataset retrieved inputs and from inputs that are retrieved from the last step of the prediction. The latter has the objective of replicating the prediction as it is achieved from a typical constitutive model. The aim is the minimisation of errors after training. The feedback and the non-feedback predictions do not produce the same results which imply that the network is sensitive towards a certain input. This is further validated by conducting a sensitivity analysis and by looking into the activation of each node for certain loading cases. Dropout and reassessing the inputs and outputs are attempted to resolve this issue but the results remain erroneous.

The second approach is to create a component based Neural Network. In this case a link is created between the function of the neural Network and typical soil behaviour. The linear elastic model is modelled with a linear activation function. In this case the network is successful in reproducing the full linear-elastic matrix. The linear elastic perfectly plastic model is modelled by connecting the linear elastic matrix with a ReLU layer as it is seen in continuum mechanics. The Neural Network accurately predicts the stress-strain relationship. And it can be used to also predict the stress path of “noisy” datasets. However, when trained with noise the signal added to the training dataset is recognised as a pattern from the Neural Network. Finally, the work hardening model does not successfully model the stress-strain relationship as it tends to exaggerate the contribution of the stress input versus the strain input. All in all, this is an effort towards the development of a Neural Network constitutive model with the final aim of producing data driven constitutive models.





# Chapter 1. Introduction

---

Nowadays, structural analysis is one of the main tools used in geotechnical engineering to determine the fitness of use for structures. From the development of Hooke's law to the first formulation of the "Finite Element Method" (Turner, Clough, Martin, & Topp, 1956) constitutive equations help relate stress with strain, the two physical quantities mostly used when describing solid behaviour. Of course, this is also expanded to Geotechnical Engineering where the solid is a geo-material and constitutive equations predict the behaviour of it. Since, constitutive models become more and more complicated and require a number of parameters with little physical correlation, the need for data driven constitutive models becomes apparent. In the following chapter the background of the project is described together with the research goal, scope of work and the outline of the thesis project.

## 1.1 Thesis Background

Constitutive equations are usually governed by the properties of a material and can be as simple as a linear relationship or more complex and account for rate of response. The constitutive equations governing soil behaviour have become increasingly complex. Many of them feature 10 or more parameters as inputs or possess parameters that are hard to understand (Beatty & Byrne, 1998; Bauer, 1996). However, no matter the complexity introduced to the model constitutive models are still mathematical models with parameters estimated from laboratory or field data. Thus, they cannot capture with full accuracy the complexity of soil behaviour. Therefore, investigating a more data-driven approach could address these issues, improving existing material models.

Nowadays, artificial intelligence tools have been gathering more attention as they can solve complex problems and are often highly efficient and accurate (Nilsson, 2009). There are a number of successful Artificial intelligence tools that vary from fraud detection, image recognition, autonomous cars to competing at the highest level in strategic game systems (such as chess and Go) (David, Netanyahu, & Wolf, 2016; Clark & Storkey, 2015). Out of these tools Neural Networks seem to be one of the most promising and largely developed Artificial intelligence tool. (Hern, 2016)

In relation to the constitutive models for soil the three most popular Artificial Intelligence methods include genetic programming, evolutionary polynomial regression and artificial nested networks (ANN). Artificial Neural Networks as stated before are largely developed and more promising than other techniques. Therefore, the thesis will be focused on applications of Neural Networks regarding constitutive modelling.

## 1.2 Proposed solution

The need for a more data driven approach concerning constitutive modelling together with the recent successful developments in Neural Networks (Brownlee, 2016; Shahin M. , 2012) gives rise to the thesis goal. This thesis attempts to assess if it is possible to reproduce various constitutive models by using neural networks and in this process investigate the process and the requirements of creating a successful Neural Network Soil Material model.

## 1.3 Research Question

From the proposed solution part the scientific question of the thesis becomes clear. The general scientific question of this thesis is "*Can Neural Networks behave as successful constitutive material models?*" From the main research question more specific sub-questions can be formulated.

- *Can Neural Network model typical behaviours captured in soil constitutive models? For example loading- unloading- reloading of soil, linear elastic behaviour and soil hardening.*
- *How will the resulting Neural Network perform with if laboratory data or data retrieved from constitutive models that exhibit more complex behaviour, are the input? For example data retrieved from triaxial tests or from constitutive models like Hardening soil model or Cam-Clay model.*

## 1.4 Scope of work

The objective of the thesis is to expand the knowledge on constitutive models created from Neural Networks and discover if they are successful in modelling typical soil behaviour. The research will consist of three main parts:

1. Creating the datasets consisting of soil data. The dataset will be used for training and validating the models. This will be done by using already validated and accepted material models (e.g. Linear-Elastic, Mohr Coulomb)
2. Create generic Neural Networks. In this case the Network is treated as a black box the main focus is concentrated on the Neural Network functions rather than the constitutive model behaviour.
3. Create Neural Networks that capture typical soil behaviours of each model. In this part attention is paid to the inner workings of the Neural Network as well as the actual behaviour of the constitutive models.

In this study the main focus is to understand how soil behaviour relates to Neural Network computations (activation functions, weights, and back-propagation) and in general understand what steps need to be followed to create a successful Neural Network Material Soil model.

## 1.5 Report Outline

In Figure 1 the outline of the report can be observed. The thesis consists of three main parts. In Chapter 2 the literature study, consists of the theoretical background of concerning Neural Networks. In Chapter 3 a generic Neural Network will be attempted. The accuracy of the model is assessed together with its generalisation ability and the final conclusions for the model. The second approach, Chapter 4, consists of component based modelling. In this case the linear-elastic model, the linear-elastic perfectly-plastic model and the linear-elastic work-hardening model are reproduced using specific parts of the Neural Network. A section with concluding remarks and summary is added to each chapter. Finally, the discussion of the thesis is made in Chapter 5 and the general conclusions of the thesis are made in Chapter 6. Finally, recommendations are made in the final part of the thesis Chapter 7.

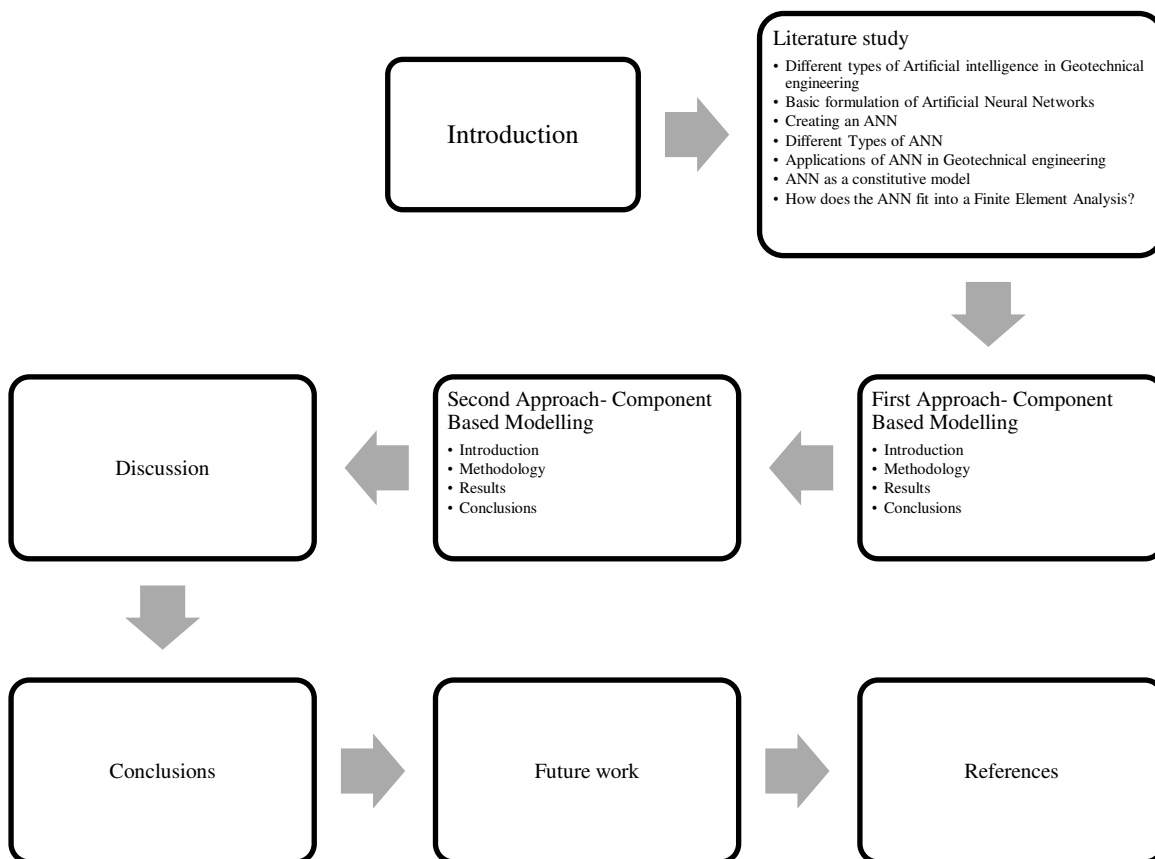


Figure 1 Reader's Guide

# Chapter 2. Literature Study

The Literature Study will focus on Neural Networks. The aim of the Literature Study is to gain an understanding on how Neural Networks work, critically review the theory which will be needed for the analysis of the thesis's results and finally use the results of previous research to identify a promising direction for future research. The first step of the literature review is an introduction to Neural Networks, thus their basic structure and computational operations. Narrowing the scope of the literature study a list of typical applications of Neural Networks in geotechnical engineering will be presented. By reviewing the papers that apply to constitutive modelling the steps of creating a Neural Network for constitutive models will be identified and finally the main types of Neural Networks found in literature will be reviewed as well. Finally, a review of how Neural Networks are used in Finite Element Analysis will be made.

## 2.1 Neural Networks- Basic Structure and Computational Operation

Artificial Neural Networks are computing systems that are inspired by the biological equivalent neural networks that are part of animal brains. These systems “learn” various tasks by comparing outputs either between them (unsupervised learning) or with a target output the user provides. In that way the Neural Network creates their own set of characteristics from the learning material they process.

### 2.1.1 Structure-The Neuron

The building blocks of this computational system (as of its biological equivalent) are the neurons. The neuron is inspired by the biological neuron, in the sense that the functions performed by them are similar. Interconnected neurons use electrical pulses to “communicate” with each other. The biological neuron has four basic components (Figure 2) the dendrites, soma, axon and synapses. The neurons are connected with each other through the synapses (of the first neuron) to the dendrites (of the second one). When the neurons synapses are triggered by an electrical signal, the signal is processed in the soma of the cell, it travels through the axon and finally the signal is released through an electrochemical contact from the synapses to the next neuron.

To be part of an Artificial Neural Network, the biological neuron (Figure 2) can be reduced to a mathematical function. The artificial neuron (Figure 3) receives one or more inputs which are usually weighted and then summed. After the summation the result is passed through a usually non-linear function called “activation function”. Typical activation functions include the “linear”, “sigmoid”, “tanh” or “ReLU” functions.

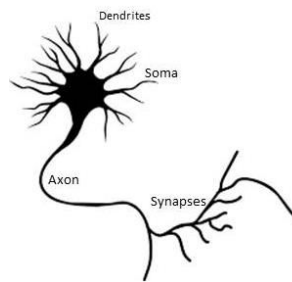


Figure 2 Structure of the biological Neuron (Davies, 2002)

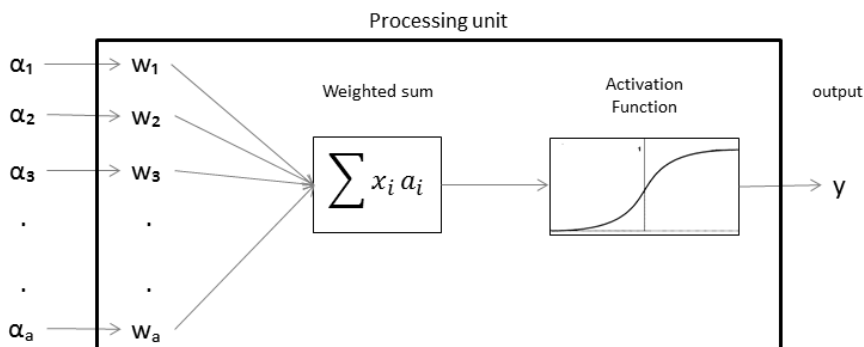


Figure 3 Processing Unit of Artificial Neuron

### 2.1.2 Structure-Connections and Full Neural Network

To create a Neural Network the neurons described in chapter 2.1.1 will be connected to each other from inputs to outputs. The Neural Network can also be defined in terms of layers. A layer of a Neural Network consists of a specific amount of neurons and are usually fully connected to the neurons of the next layer. That means that each neuron of a layer will be connected to each neuron of the adjacent layer (Figure 4). Typically, there are at least three layers of neurons in each Network. The first is the input layer which is defined as the number of inputs of the Neural Network. The last layer is called the output layer and the number of neurons corresponds to the number of outputs of the network. The intermediate layers are the hidden layers of the network and are selected arbitrary or by using techniques or rules discussed in the following chapters.

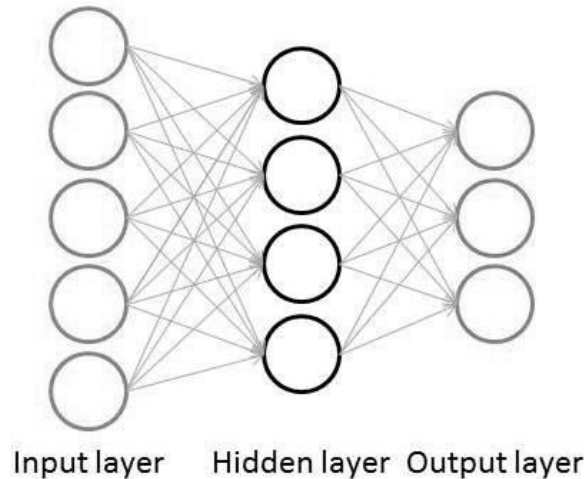


Figure 4 A typical structure of a Neural Network

### 2.1.3 Computational Operation- Feed Forward Neural Network

The Neural Network operates based on the concept that activations in one layer determine the activations of the next layer. This process can be explained through a simplified example and then with added complexity. If we consider a 3 layered network with one node as an input, two nodes as a hidden layer and one node as output the following process will be followed (Figure 5):

- The weights are initialised randomly. After that the inputs are fed into the network.  $([a_1 \ a_2])$
- The inputs are weighted according to weight corresponding to each of the neurons. The weighted input is then summed and a bias is added to them.  $(w_1 a_1 + w_2 a_2 + b_1)$  The bias acts as a threshold value and thus it allows the activation function to be shifted left or right while the weights determine the “steepness” of the activation function. The result of the above will be passed through an activation function (e.g. sigmoid or linear)  $(f(\sum_{i=1}^2 w_i a_i + b_1))$ .
- Then the result will be weighted once again and the bias and activation function of the last node will be added to that to produce the last output.  $f(f(\sum_{i=1}^2 w_i a_i + b_1))w_3 + b_2$

Moving forward from the example the general mathematical approach will be considered and formulated. Each input of the input layer is connected to each neuron of the hidden layer and so on until the last output layer. All of the inputs are given a certain activation that will be a number usually scaled between 0 and 1.

$$[\alpha_1 \ \alpha_2 \ \dots \ \alpha_n] \tag{1}$$

Where n is the number of nodes in the input layer. A weight is then assigned to each one of the connections between each of the neurons in the hidden layer and the neurons from the input layer. The weighted sum of all of them is computed according to these weights. Thus, for each neuron in the hidden layer:

$$\alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_n w_n \tag{2}$$

Then a bias will be added to the results of equation 2. The bias is added so that the activation function can become active only after a certain threshold value is reached. Therefore, the bias determines when the neuron remains inactive.

$$\alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_n w_n + b_1 \quad (3)$$

This weighted sum is passed through an activation function  $f$  (which is usually a sigmoid). In the case of the sigmoid the output (equation 3) is between 0 and 1 and can be considered as a measure of how positive the weighted sum is.

$$f(\alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_n w_n) \quad (4)$$

If we consider the full architecture of the Neural Network in Figure 6 the full equation can be formulated. The activations of each layer are organised as a vector. Where  $\alpha(0)$  is referred to a specific layer of the Neural Network.

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad (5)$$

The weights are then organised as a matrix. Where each row of the matrix corresponds to the connections between one layer and a particular neuron in the next layer.

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad (6)$$

Then the bias is added to equation 6.

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (7)$$

Then equation 7 is wrapped around the selected activation function.

$$f \left( \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \right) \quad (8)$$

Equation 8 represents that the activation function will be applied to each specific component of the resulting vector inside it. So the final feed-forward Neural Network equation will be:

$$\alpha^{(1)} = f_1(\mathbf{W}^{(0-1)}\alpha^{(0)} + \mathbf{b}^{(1)}) \quad (9)$$

Finally for one hidden layer architecture the equation regarding the outputs will be:

$$\alpha^{(2)} = f_2(\mathbf{W}^{(1-2)}f_1(\mathbf{W}^{(0-1)}\alpha^{(0)} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \quad (10)$$

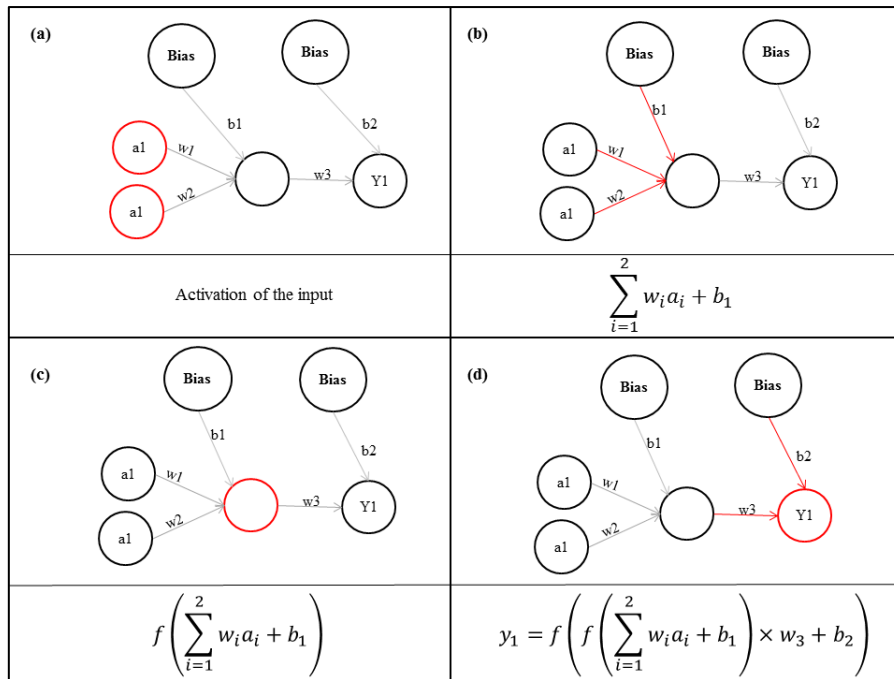


Figure 5 Simple Neural Network example Feed-Forward method

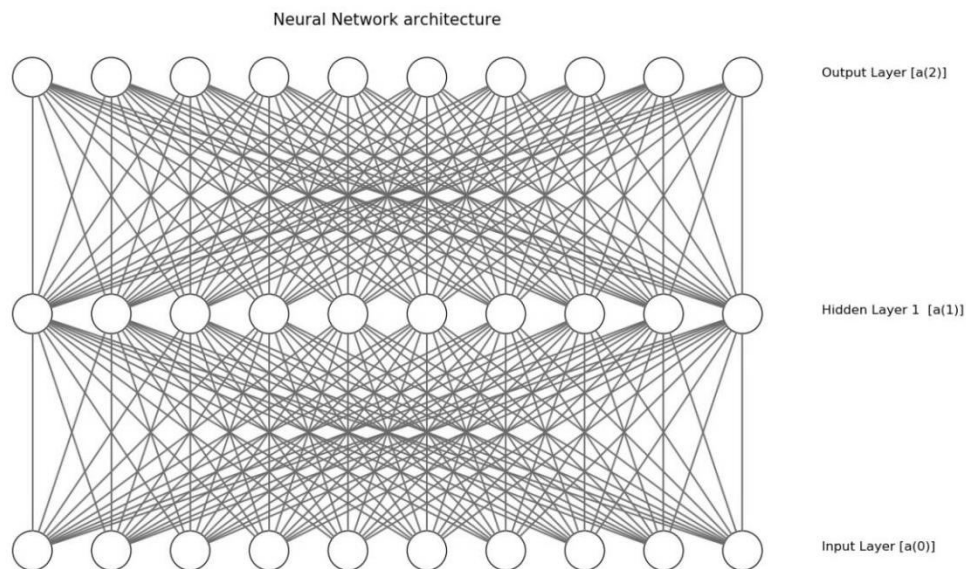


Figure 6 Typical structure of neural Network

### 2.1.4 Computational Operation- The back propagation method

Training Neural Networks can be achieved with supervised or unsupervised learning. The difference between them is that with supervised learning the network will try to minimize the errors between inputs and outputs by adjusting the weights. Whereas in unsupervised learning the Neural Network does not have the ability to calculate errors and thus the Neural Network will model the underlining structure of the data. In Geotechnical applications Neural Networks are trained by using the method of supervised learning since the outputs of the network are strictly defined.

As stated earlier, with the supervised training the Neural Network will try to improve its performance by minimizing errors between the predicted and the actual output. Backpropagation is a method used in Neural Networks to calculate the gradient of errors with the intention of adjusting the weights of the network. (Goodfellow, Bengio, & Courville, 2016). To calculate the gradient of the errors a function needs to be defined. This function is typically called “cost function” and it calculates the difference between the network output and the expected output. A simple form of this function can be seen in equation 11. Where  $y(x)$  represents the output

of the target function and  $y'(x)$  the prediction of the neural network,  $n$  is the number of training samples that the training examples are going to be averaged to.

$$E = \frac{1}{2n} \sum_x \|y(x) - y'(x)\|^2 \quad (11)$$

The cost function is responsible for adjusting the weights according to its results. The first step is to calculate the partial derivative of the error with respect to each weight in the network. The gradient of each weight is essentially an indication of how much the change of a specific weight is affecting the final cost function.

$$\nabla E = \left( \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_N} \right) \quad (12)$$

To finally update the weights a learning rate  $\lambda$  is used. The negative notation in the gradient is used because the aim of the training is to decrease the function  $E$ . The updated weights are derived by equation (13).

$$w_{\text{new}} = w_{\text{old}} - \lambda \nabla E \quad (13)$$

If the cost function was represented in an x-y space, where  $x$  is the error and  $y$  are the weights, we would see that is not a monotonic function and in that way it possesses various local minima (Swirszcz, Czarnecki, & Pascanu, 2017). When the training of a neural network begins the weights are typically initialised randomly. This might result in the loss function getting stuck in a local minimum and thus the resulting weights will not be the smallest the network can achieve. (Figure 7)

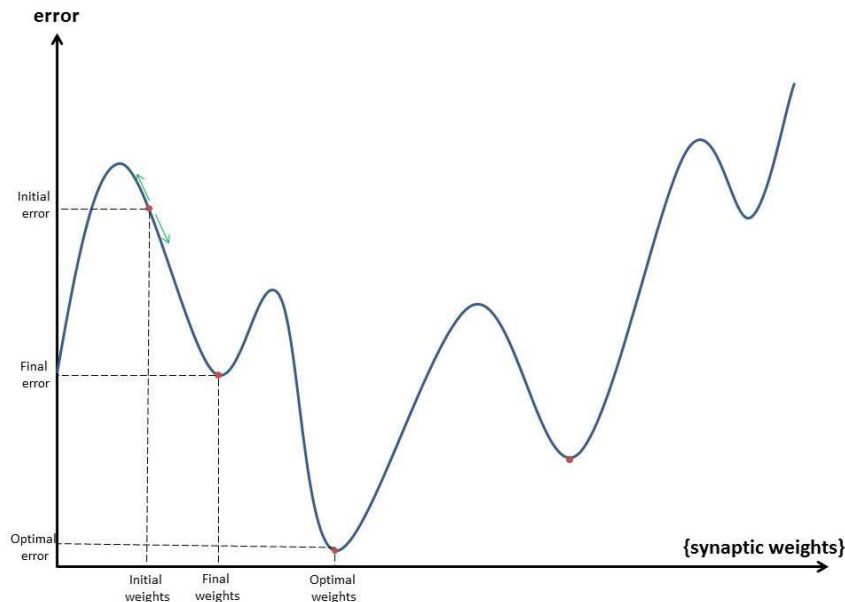


Figure 7 Local minima problem

### 2.1.5 Computational Operation- Activation functions

In the previous chapters the term activation functions was quoted. In this chapter the use of activation functions in Neural Networks is explained. As it is implied in chapter 2.1.3 activation functions define the output of a neuron for a given set of inputs. The first activation functions were a type of step functions where the only possible output is either 0 or 1 (in that way “ON” or “OFF”). Thus, if the input value was larger than a threshold the output would be 1 and if it was less than the threshold the value is zero.

However, if the output of the Network should not be binary the step activation function is not useful. In this case a linear activation function can be used. By using this type of activation function the aim of the network is to model a straight line where the activation is proportional to the input. However, the problem with this activation function is that if a gradient descent algorithm is used then the derivative of this function will be a constant value. That implies that regardless of the layers of the network the final activation function of the last layer is a linear function of the inputs. That means that if the network contains multiple layers they can be replaced by a single layer because the ability of stacking layers is lost by the linearity of the activation functions. (Jordan,








1995). However, if the final function modelled is linear then a network composed of linear activation functions is useful.

In most Neural Network cases the activation functions are used to introduce non-linearity to the network. A non-linear activation function is able to compute problems with using only a small amount of neurons. A sigmoid function is the next choice. This type of function is non-linear but also has the ability to act as a step function since after a certain number the result of the activation will either be 1 or 0. However, if the steepness of the curve is large that will mean that small changes in the input values will result in large changes in the output values. In addition, towards either end of the sigmoid function the values of the outputs tend to respond less to changes of the inputs. The gradient of that region is, thus, vanishing. This issue is called the vanishing gradient problem. In this case the network’s learning process is stopped or slowed down.

The final activation function used in this thesis is the ReLU. This function is non-linear, as values smaller than zero will return a zero output. ReLU is usually less computationally expensive as it involves simpler mathematical operations. However, ReLU is governed by the “dying ReLU problem”. In this case because of the horizontal line it is possible that the gradient will return values of zero. That means that the specific weight connections that are not “fired” will stop responding to the error variation and so they will stop training. The leaky ReLU is a variation of this activation function that deals with this problem. In that case the line is slightly inclined and thus the gradient will not be zero.

Table 1 Activation Functions

Name	Plot	Equation	Range
Linear		$f(x) = x$	$(-\infty, +\infty)$
Step Function		$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	(0,1)
Sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$	(0,1)
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$(0, +\infty)$
Leaky rectified linear unit (Leaky ReLU)		$f(x) = \begin{cases} 0.001x, & x < 0 \\ x, & x \geq 0 \end{cases}$	$(-\infty, +\infty)$

## 2.2 Applications of Neural Networks in Geotechnical engineering

A lot of successful studies exist that tackle geotechnical problems. In this section those studies are summarised to determine popular Neural Network approaches to Geotechnical problems. Some of the studies are concerned with **pile foundations**. An attempt is made by W.T. Chan (1995) to find an alternative pile driving formula using Neural Networks. However, more successful applications are focused on recovering the axial capacity of pile foundations (Shahin M. , 2010) , the lateral load capacity in clay soils (Das & Basudhar, 2006) in these cases the Neural Networks used have simple architectures of one hidden layer and multiple inputs of soil parameters. The next applications calculate the efficiency of pile groups (Hanna, Morcou, & Helmy, 2004) and

the pullout capacity of anchors (Shahin & Jaksa, 2003) using fuzzy logic<sup>1</sup>. Those applications proved to be equally or even more accurate than the formulas in the existing literature.

For determining the **settlement of foundations** in most cases a simple Neural Network is not accurate enough to model this complex non-linear behaviour. In some cases Recurrent Neural Networks (RNN) are used to produce more accurate predictions (Shahin M. , 2014). In this case CPT data are used as an extra input and as the author suggests “*the RNN can capture highly non-linear behaviour*”. Neuro- fuzzy systems are also used to predict the load-settlement curves of strip foundations (Provenzano, Ferlisi, & Musso, 2004) and to predict the ultimate bearing capacity of shallow foundations (Padmini, Ilamparuthi, & Sudheer, 2008). The fuzzy Neural Networks appear to be quite robust and respond well to experimental data in this case.

Studies for determining **slope stability** also exist. In the case of S.Kumar Das (2011) the Neural Network is used as a classification algorithm. The aim is to classify the slopes between failure (represented by the number 0) and safety (represented by the number 1). However, in other studies the stability problem is approached with uncertain quantities. In this case the Neural Network approximates the limit state function which is linked to a reliability method the final outcome of the framework is the failure probability (Cho, 2009). Ferentinou (2007) aimed to uncover the inner working of the Neural Network that lead to a successful prediction with the technique of self-organizing maps implemented together with the Neural Network.

For the **determination of soil properties** simpler Neural Networks can be implemented. For example the unsaturated shear strength can be determined by only one hidden node layer with six inputs of sand, clay or silt friction, void ratio, compacted  $w(\%)$ , cohesion ( $c'$ ) and friction angle (Lee, Lee, & Kim, 2003). The same idea is used by S.Kumar Das (2008) to determine the residual friction angle by multiple soil parameter inputs. Finally, a PHD study is made by Obrzud (2009) to extrapolate constitutive parameters from in-situ soil data which is also quite successful.

To access the **liquefaction potential** of soils M.H.Baziar (2007) used a large database of laboratory cyclic data as well as cyclic liquefaction tests to predict the amount of strain required to trigger liquefaction.

Studies that model **stress-strain relationships** with Neural Networks also exist. Most of the networks are recurrent with extra inputs of soil parameters (relative density, confining pressure etc.). The implementation is quite simple and the results have a relative good fit. (Banimahd, Yasrobi, & Woodward, 2005; Zhu, Zaman, & Anderson, 1998; Romo, García, Mendoza, & Taboada-Urtuzuástegui, 2001; Penumadu & Zhao, 1999; Johari, Javadi, & Habibagahi, 2011; Bamdad & Habibagahi, 2003)

The applications for **constitutive modelling** start with J. Ghaboussi (1991). In this study the auto progressive training of the Neural Network is implemented. The Neural Network is quite successful at modelling the constitutive behaviour since the training is achieved by the global load-deflection response measured in a structural test. The network is attached to an iterative non-linear finite element analysis so that the stress – strain relationship can be gradually extracted to train the Neural Network. This idea is later expanded with the addition of history points to the inputs and outputs of the Neural Network (Ghaboussi & Sidarta, 1998) which increases the accuracy of the prediction. Chaboussi and Sidarta (1998) use the nested adaptive neural networks in an auto progressive training with data from a wide variety of non-linear stress paths. This study reveals the effect on predictions when testing data what were not included in the training. Another technique is later developed by Amir H. Gandomi (2014) it is called the SelfSim method which is used for the inverse extraction of non-linear material behaviour. The approach of Jorg F.Unger (2009) is to use different applications of Neural Networks to help with implementing a multiscale analysis on reinforced concrete. Specifically, the Neural Network is used as a material model between the concrete and the reinforcement. However, to apply this method, modifications had to be made to the Netwon- Raphson iteration in the extrapolation and superposition of the elastic material. D. Stefanos (2015) focuses on the training of a simple Neural Network with a 2D Cartesian strain vector as an input and the Cartesian stress vector as an output. The test data in this case are created in PLAXIS using the Hardening Soil model.

On more application of Neural Networks is to test the behaviour of **diaphragm walls** (Kung, Hsiao, Schuster, & Juang, 2007). Specifically, the output of the network is the deflection of the diaphragm wall. In this case several variables such as excavation depth, system stiffness, excavation width, shear strength; effective strength and Young's modulus are used as input to the network. The Neural Network is validated against in situ data.

---

<sup>1</sup> “Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false.” (Novák, Perfilieva, & Mockor, 1999)

For calculating the **stress history** Pradeed U.Kurup (2002) developed a Neural Network tool to calculate the OCR values by using inputs of cone resistance ( $q_c$ ), total overburden pressure ( $\sigma_w$ ), and several pore pressure inputs on the cone and the hydrostatic pressure. Finally, as a result the OCR output values have a good accuracy.

From reviewing various Neural Network applications it can be concluded that there are 2 types of approaches when dealing with Geotechnical problems and using Neural Networks. Firstly, geotechnical problems that have as an output one value as a prediction and are not time depended can be easily trained by a Neural Network as long as enough parameters are set as inputs. In the case that there are not enough parameters, the Neural Network has the ability to be attached to probabilistic computational tools. The second type of Neural Networks architecture has as a goal to model soil stress-strain relationships. This type of relationship requires the implementation of an elaborate general framework. For these applications, usually the Neural Network is linked to a finite element analysis which helps with fine tuning the weights of the Neural Network. However, it can be observed that the link between the failure and success of the Neural Network model with the inner workings of it is rarely made. Thus, the role of each node in the final prediction is not investigated. This will be an approach investigated in this thesis. Finally, the versatility of the Neural Network tool and the amount of applications it can be successfully implemented build a strong case in favour of the scientific question.

## 2.3 Creating an Neural Network

As stated earlier many consider Neural Networks to be black boxes (Benitez, Castro, & Requena, 1997). Therefore, many perceive the choices in developing a Neural Network are made arbitrary. To resolve this common misconception the literature study at this point will focus on the steps followed to create a Neural Network as they are defined in literature. The main steps in developing the Neural Network can be summarised in Figure 8 (Maier & Dandy, 2000b). The main steps of the Neural Network are described in this section. However, many of them are excluded from the following chapters (e.g. Choice of optimization method, Model inputs) as they are already defined in literature and they are beyond the scope of the project.

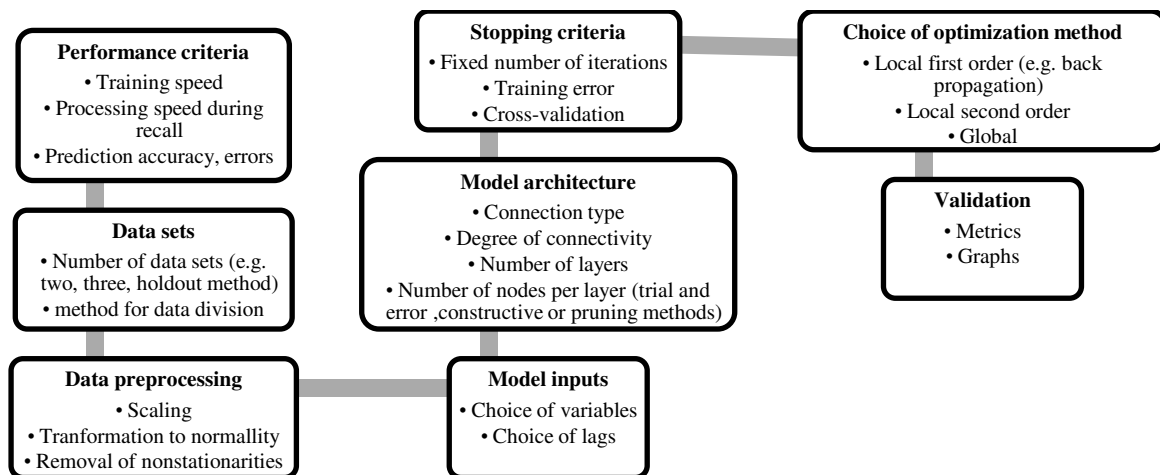


Figure 8 Main steps in Neural Network development

### 2.3.1.1 Training and Testing Dataset Determination

The data sets created need to be further subdivided into subsets before they are used. The two main subsets are the training and the testing subset. (Shahin & Jaksa, 2004b). The training one, as the name implies, will be used to train the Neural Network. Thus, it will be used to adjust the weights to the optimum values. The testing dataset consists of samples that the network is not trained on. This dataset will be used for validating the final trained neural network.

Usually 80% of the data are assigned to the training subset and 20% to the validation. However, some studies suggest that the data should be subdivided into three categories: training, testing and validation. The training subset should contain 55% percent of the data and is used for adjusting the connection weights. The testing subset contains 25% of the data and is used to check the model in different stages, and determines when to stop the training process to avoid overfitting. The validation set contains 20% of the data and is used to check the performance with relation to the environment. (Shahin & Jaksa, 2004b). These percentages are arbitrary but the main concept behind the data division is that the training set should be representative of the data that the network will be tested on (Maters, 1993). The dataset range used for the training should not be smaller than the data set used for the actual application of the neural network. Thus, extreme values of the data should be located in this testing dataset.

However, if the training error is too small the model might be overfitting. In this case the model “memorises” specific patterns in the training subsets that are not connected to the general behaviour of the dataset. Therefore, a separate testing set needs to be created to test if the Neural Network avoided overfitting. This is the validation subset as mentioned above. An example of overfitting data is presented in Figure 10. As the error in the training data starts decreasing the error in the testing data is increasing (Bamdad & Habibagahi, 2003). There are certain rules of thumb that exist to avoid overfitting networks. Firstly, the number of training samples should be larger or equal to the number of weights. (Rogers & Dowl, 1994). However, others claim that overfitting does not occur if the number of training samples is at least 30 times the number of free parameters and they define the number of weights as  $w=(I+1)H+(H+1)O$ . Where H is the number of neurons in the hidden layers, I is the number of inputs and O the number of outputs. (Amari, Murata, Muller, Finke, & Hua Yang, 1997)

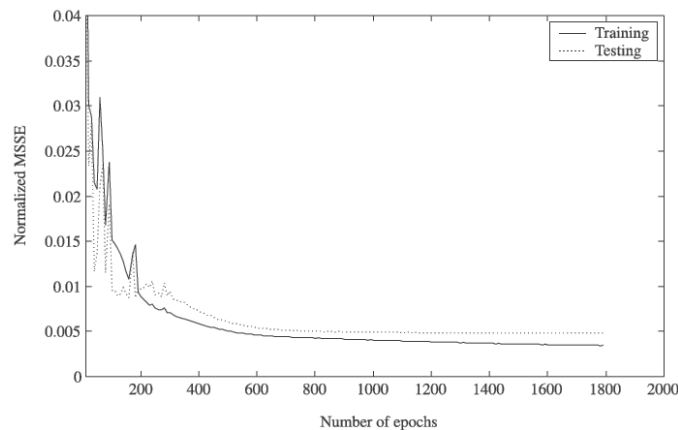


Figure 9 Performance during training (Bamdad, 2003)

### 2.3.1.2 Data Pre-processing

Scaling the data is usually determined by the function the Neural Network is supposed to model. Some typical methods of rescaling include:

- Normalisation: here the data are divided by a norm of the data. This technique aims to make the length of the data equal to one. Typically normalising the data refers to rescaling by the minimum and range of the data vector so that all the samples lie between 1 and 0.
- Standardizing: in this case the data vector is subtracted by a value that acts as a measure of location and then it is divided by a value that acts as a measure of scale.

Generally many cases do not require scaling of the data as it can lead to loss of information, especially with experimental data. However, in other cases the scaling of the input data vector is a good idea as Neural Networks tend to favour larger sample values. Thus, this leads to poor generalisation as the weights are mostly fitted to them. This is also the case for multiple inputs in a Neural Network. For example, if one input has a range between 0 and 1, while another ranges between 0 and 100.000 then the Network will become partial to the second input (Sarle, 1999). It is also stated that the “input/output dataset has to be distributed around the midpoint of the interval in order to allow proper training of the neural network” (Koprinkova & Petrova, 1999).

### 2.3.2 Determination of the Neural Network’s architecture

The determination of the model’s architecture is achieved by selecting the optimum number of layers and nodes in each of the layers. Most of the applications for Geotechnical engineer in literature use two hidden layers. In general, for neural networks it is suggested that the first layer identifies the local features of the inputs and the second layer the global ones.

Then the number of hidden nodes in each layer should be defined. In many studies the trial and error approach is used for determining the model architecture. The model is trained using 1, 2, 3...  $2I+1$  hidden layer nodes (I is the number of the input variables). Note that the number  $2I+1$  is the upper limit need to map any continuous function for a network with I inputs (Caudill, 1988). An example is presented in Figure 10. In this case the number of hidden nodes selected for the driven pile is 2 as it is considered to be the optimum and for drilled shaft are 3 nodes (Shahin M. , 2010). It is also important to keep the hidden nodes to a minimum. As it is shown in literature a large number of connection weights in a neural network will increase the likelihood of the model to overfitting and reduces the chance of generalization (Shahin, Jaksa, & Maier), (Figure 12). In a different study using Neural Networks to describe unsaturated soils the optimum number of nodes is found with a trial

and error method but by decreasing the number of neurons from a high number while checking the error in the network (Bamdad & Habibagahi, 2003). In the trial and error method different architectures are trained for a large number of epochs. After an increment of epochs, the testing dataset is fed into the Neural Network and the testing error is calculated. The optimum architecture can be found from this process as it can be seen in Figure 11 (Penumadu & Zhao, 1999).

However, systematic approaches exist that can determine automatically the network's architecture. The adaptive method of architecture determination is one of the automatic methods (Figure 13). The model starts with an arbitrary and small number of nodes. During training, new nodes are added to the hidden layers and thus new connection weights are generated. The training process continues so that the new connections obtain the optimum weight values. While this process runs the old weights remain unchanged. Additional training cycles are then performed where all the connection weights are allowed to change (Sidarta & Ghaboussi, 1998).

Another criterion used for determining the optimum number of hidden neurons is the Bayesian information criterion (BIC) (Jiang, Mahadevan, & Yuan, 2016). In the end the criterion will select the simpler model (so the model with the least parameters) that will also have a good fit of the data. The function that defines the criterion is:  $IC(J) = -2D(\theta) + n_p \log(N_{max})$  (Raftery & Kass, 1995).  $\Theta$  represents all the parameters estimated by the model. The  $n_p$  is the total number of parameters that will be updated in the model (e.g. the weights).  $D(\Theta)$  is the observed data log-likelihood function with regard to  $\Theta$ . The model with the lowest BIC will be preferred.

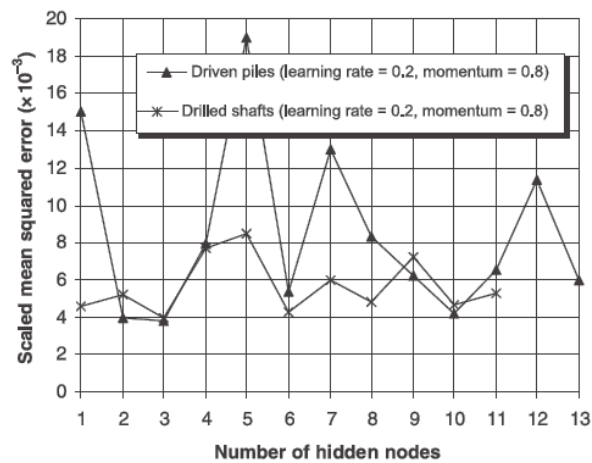


Figure 10 Effects on number of hidden nodes on performance of ANN model (Shahin M. A., 2010)

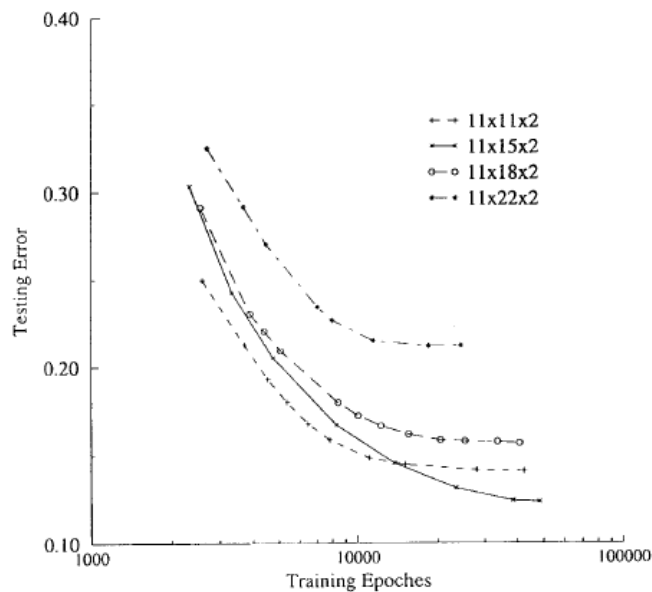


Figure 11 Variation of error associated with a testing data set (Penumadu & Zhao, 1999)

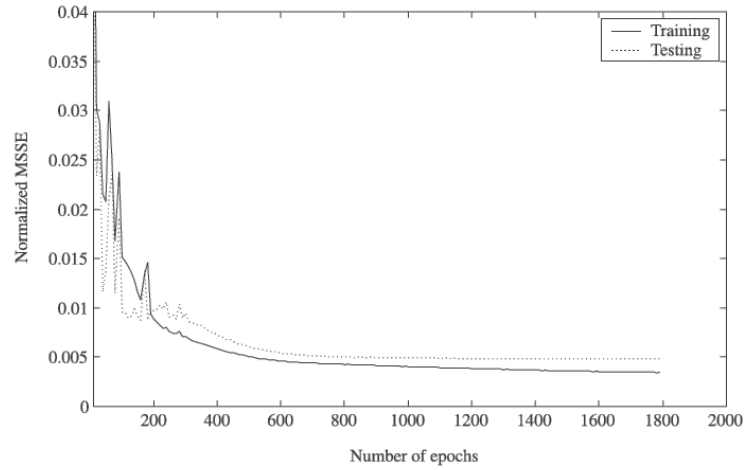


Figure 12 Performance during training (Bamdad, 2003)

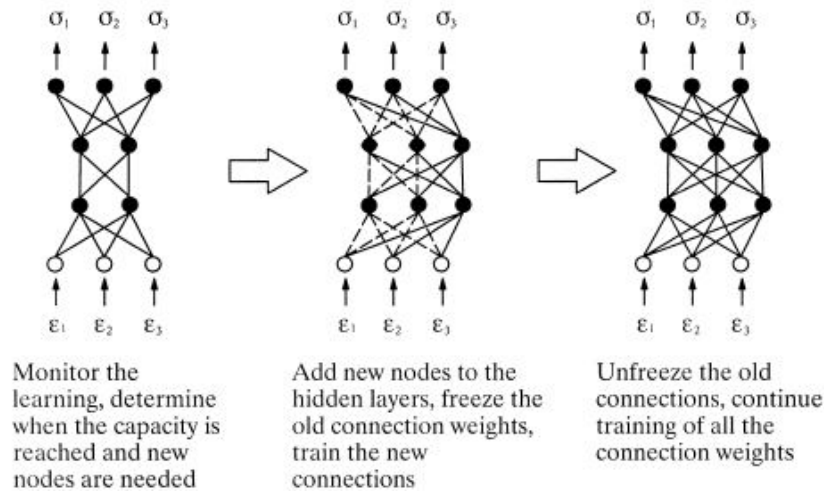


Figure 13 the procedure for the adaptive evolution of the neural network architecture during training of the neural network material method. (Sidarta & Ghaboussi, 1998)

### 2.3.3 Validation

After the training process is finished the model must be validated. The model is tested for its generalisation abilities within the limits of the training data. This is tested by running the model with the validation subset. The main criteria used to quantify the prediction performance of the model, are the coefficient of correlation ( $r$ ), the root mean squared error (RMSE) and the mean absolute error (MAE). The coefficient of correlation is a measure of relative correlation and goodness-of-fit between predicted and observed data. With the RMSE error the large errors receive greater attention while the MAE error is an absolute measure of error.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (14)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2} \quad (15)$$

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j| \quad (16)$$

The robustness of the model can be tested by performing a sensitivity study. In that way the response of the model can be checked for different inputs. Hence it is checked if the model responds according to the underlining physical process. (Shahin, Maier, & Jaksa, 2005) Another way to test the robustness of the Neural

Network is with the connection weight approach. The concept is that in order to check the generalisation of the Neural Network, the relationship of the generated and the relationship between input-connection-output should be quantified. This can be achieved by determining the strength and direction of the connection weights between them. Finally the Relative Contribution of each input can be calculated as a percentage. From the equivalent physical relationships with the output can be evaluated (Kingston, Maier, & Lambert, 2005). The metric of Relative Contribution is explained in detail in Appendix B.

Apart from the Relative Contribution another metric is the Relative Importance. This method proposed by Garson (1991) and later modified by Goh (1995) partitions the Neural Network's weights to determine the relative importance of each input variable in the Neural Network. However, in the Garson's algorithm the absolute values of the connection weights are used to calculate the relative importance. This implies that the direction of the relationship between inputs and outputs is not calculated. In addition, the Garson's algorithm can only be used for one hidden node layer. The calculation of the Relative Importance is explained in Appendix C.

Usually the Validation process also includes graphs so that the reader can easily identify if the model was successful in the end. For example in the measured and predicted values for each set of inputs-outputs are plotted. The data should be located close to the 1:1 line or in the 10% lines (Figure 14), (Shahin M. , 2010).

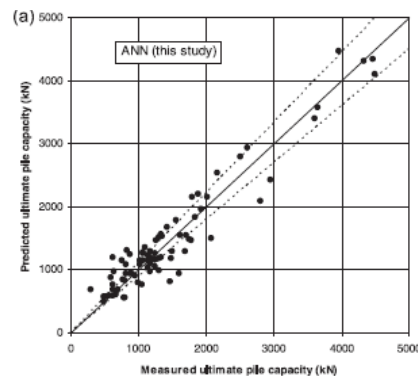


Figure 14 Performance of the ANN driven piles model (Shahin M. , 2010)

### 2.3.4 Neural Network Callbacks

A callback is defined as “a set of functions to be applied at given stages of the training procedure of the Neural Network”. A callback is usually used to view internal states and statistics of the model during training. Thus, it helps to define if the Neural Network model is overfitting or if the actual problem is ill-posed.

#### 2.3.4.1 Early Stopping

Early stopping is a form of regularization with the main purpose of avoiding overfitting. The goal of training a neural network is to obtain the optimal generalisation performance. However, if the network is overfitting the model will perform very accurately with data from the training dataset but poorly with data from the general training dataset. (Morgan & Bourlard, 1990)

The concept of early stopping is that training will be stopped in the optimum level of error for the training and validation data, thus, resulting in a network that has achieved generalisation. What is needed in this case is a predicate that will define when the training is stopped. This is defined as the stopping criterion. A typically used stopping criterion is the loss of the validation data.

#### 2.3.4.2 Dropout

Dropout is a technique for preventing the overfitting of neural networks. When the training data are limited, noisy and have complicated relationships, the Network is more prone to overfitting during training. All of these features exist in soil data.

The idea behind the Dropout method is that it combines exponentially many different neural network architectures efficiently. The term “dropout” refers to the dropping out connections between nodes in a neural network. The removed connections are temporary removed from the network (Figure 15). The choice of which connections are dropped out is made randomly.

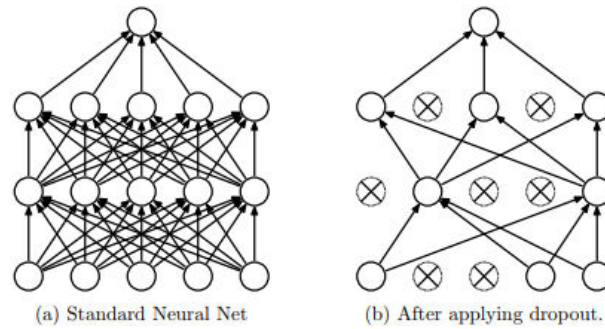


Figure 15 Dropout Neural Net Model. (a) A standard neural net with 2 hidden layers. (b) An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)

## 2.4 Different Types of Neural Networks

Two main types of Neural Networks are observed in the literature for Geotechnical applications, apart from the simple one described in chapter 2.1, the Nested Adaptive Network and the Recurrent Neural Network.

### 2.4.1 Nested Neural Networks

The material data can be described as a nested structure. (Sidarta & Ghaboussi, 1998) Modelling with a Nested neural network (NNN) allows the user to take into consideration history point what account for the path dependence of the material behaviour. This allows for a stepwise method of building and training the Neural Network to represent the complex material behaviour. The NNN consists of several modules. A base module is first created to represent the lowest function space in the material structure. The model is augmented by adding modules thus forming a higher level NNN. The process of adding modules can be theoretically continued indefinitely. All the modules represent a multi-layer feed-forward neural network. The process of creating the NNN can be observed in Figure 17. In Figure 17 it is observed that when the new module is added the weights of the previous modules are frozen as the new weights are adjusted.

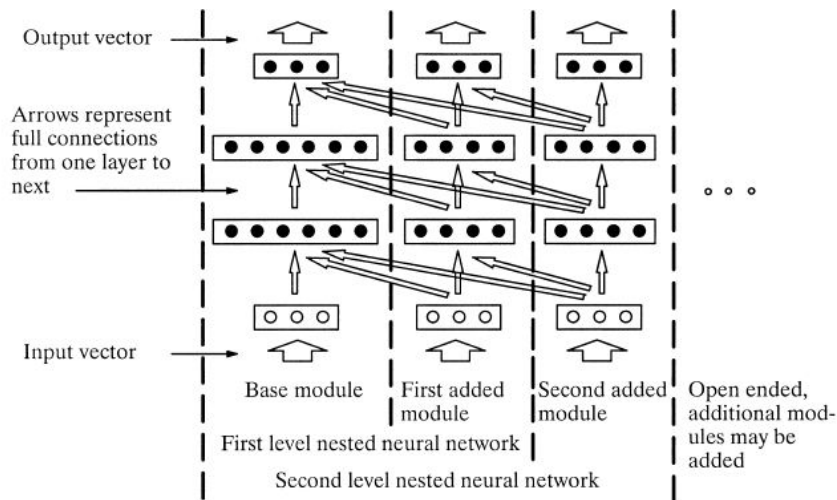


Figure 16 Symbolic representation of a typical nested adaptive neural network (Sidarta & Ghaboussi, 1998)



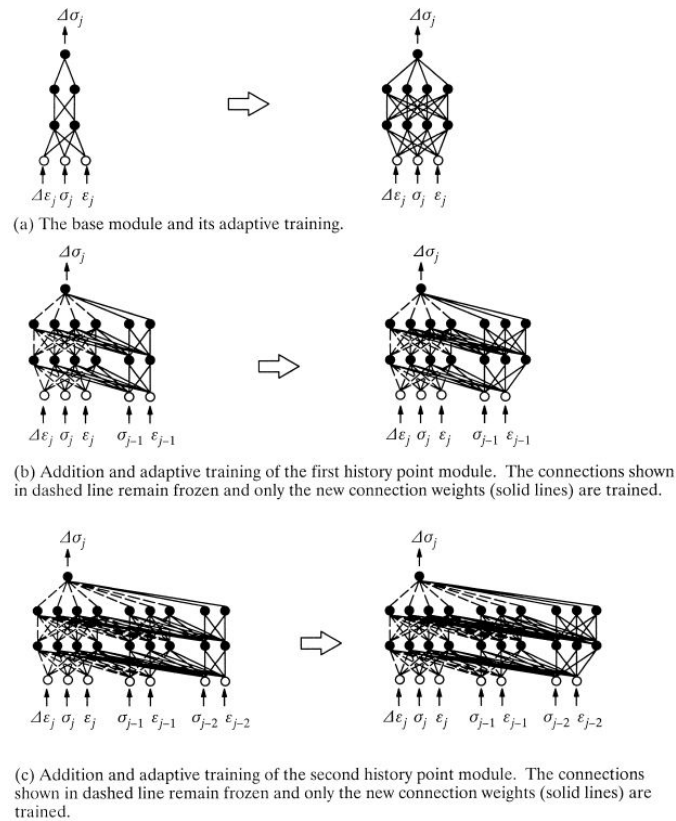


Figure 17 the evolution and training of a typical nested adaptive neural network material model with two history point modules. (Sidarta & Ghaboussi, 1998)

### 2.4.2 Feedback/Recurrent Neural Networks

Several studies have used the Recurrent Neural Network (RNN) method for Geotechnical applications. (Bamdad & Habibagahi, 2003), (Shahin M. , Load–Settlement Modeling of Axially Loaded Drilled Shafts Using CPT-Based Recurrent Neural Networks, 2014), (Johari, Javadi, & Habibagahi, 2011), (Lefik, Some aspects of application of artificial neural network for numerical modeling in civil engineering, 2013), (Najjar & Huang, 2007), (Ellis, Yao, Zhao, & Penumadu, 1995), (Ghaboussi & Jamshid, Neural network constitutive model for rate-dependent materials, 2006), The RNN has two sets of input neurons: the plan units and the current state units. At the beginning of the training process the first pattern of the input data is presented to the plan units while the current state units are set to zero. Then the first training is conducted for the first set of data. The output will be feed back into the model as a current state unit. An example of such a network can be seen in Figure 18.

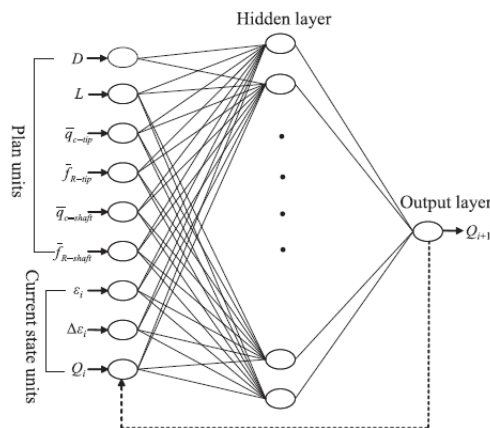


Figure 18 Architecture example of a RNN (Shahin M. , 2014)

## 2.5 Neural Networks for Constitutive models

From reviewing the Neural Network applications in the last chapter it can be seen that successful applications of Neural Network constitutive models exist in the literature. The reasons behind Neural Network being a promising alternative are that: (Pernot & Lamarque, 1999)

- In the start of the program no assumptions are made for the constitutive model.
- It can solve the problem of constitutive law inversion.
- Experimental data can be used directly to form the constitutive model.
- The model can be retrained when new data become available.

These statements seem to reinforce the initial scientific question. However, to construct a methodology for building a Neural Network constitutive model more insight has to be extracted from specific successful constitutive model cases in the literature. The chapters that follow highlight important parts that make these Neural Networks successful as well as the actual architecture of each Neural Network.

### 2.5.1 Recurrent Neural Networks

The first and most simple way to model constitutive stress-strain soil behaviour is through a Recurrent Neural Network. Main issues that must be addressed when constitutive behaviour of soil is modelled are:

- Strain increments are always feed back to the model
- Avoiding overtraining of Neural Network
- The procedure to find the optimum size of a hidden layer

Recurrent Neural Network (RNN) is largely used in stress-strain definition. The current stress and strain levels have a large influence on the next stress-strain state. In addition, some studies use additional soil parameters as inputs (for example void ratio or confining pressure) to have more accurate results. (Penumadu & Zhao, 1999), which is common practice among the RNN modelling stress-strain behaviour (Banimahd, Yasrobi, & Woodward, 2005). Another problem addressed by Lefik in 2003 is the size of the increments in the training data. He notes that they should be as small as possible. (Lefik & Schrefler, 2003). In another study (Penumadu & Zhao, 1999) the data used came from laboratory data for sand and gravel. The data were first divided into sand and gravel datasets depending on their grain sizes. The sand database was further divided into sub-databases depending on the value of their confining pressure. However, most of the RNN applications do not address the “black-box problem<sup>2</sup>” so the connection between input and outputs remains unknown. However, it is proposed that a sensitivity analysis will resolve this problem (Banimahd, Yasrobi, & Woodward, 2005).

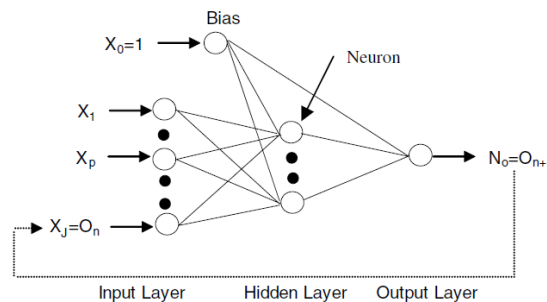


Figure 19 Typical recurrent ANN based on soil constitutive model (Banimahd, Yasrobi, & Woodward, 2005)

### 2.5.2 Nested Neural Network

The Nested Neural Network is described in chapter 2.4.1. A Neural Network material model will not produce a material stiffness matrix. When the Neural Network is implemented in a Finite Element (FE) code then this is a disadvantage. Hashash (2004) addresses this problem by considering that the stiffness matrix can be extracted from the relationship between stresses and strains (Equation 17) (Hashash, Jung, & Ghaboussi, 2004).

$$\frac{\partial^{n+1}\Delta\sigma_i}{\partial^{n+1}\Delta\varepsilon_i} = \frac{S_i^\sigma}{S_i^\varepsilon} \beta^3 \sum_{k=1}^{NC} \left( \left\{ \left( 1 - ({}^{n+1}\sigma_i^{NN})^2 \right) w_{ik}^{GC} \right\} \times \left[ \sum_{l=1}^{NB} \left\{ \left( 1 - ({}^{n+1}C_k)^2 \right) w_{kl}^{CB} \right\} \left\{ \left( 1 - ({}^{n+1}B_l)^2 \right) w_{lj}^{BE} \right\} \right] \right) \quad (17)$$

<sup>2</sup> In science, computing, and engineering, a black box is a device, system or object which can be viewed in terms of its inputs and outputs (or transfer characteristics), without any knowledge of its internal workings.

The stiffness matrix presented in Equation 17 can be used in FE analysis code for implicit methods like Newton-Raphson. In that way the material model constructed from a NANN is not a “black box” model.

- $S_j^\sigma$ : State variable for stress. The resulting stress is calculated from the multiplication of the neural network results with this value.  $\sigma_i = S_i^\sigma \sigma_i^{NN}$  where  $-1 < \sigma_i^{NN} < 1$ .
- $S_j^\varepsilon$ : State variable for strain. This is used as a scaling factor for the strain.  $\varepsilon_i^{NN} = \varepsilon_i / S_i^\varepsilon$  such that  $-1 < \varepsilon_i^{NN} < 1$ .
- $\beta$ : Constant for the sigmoid function tanh.
- $w_{kl}^{CB}$ : Connection weights between 1<sup>st</sup> hidden layer node  $B_l$  and 2<sup>nd</sup> hidden layer node  $C_k$ .
- $w_{lj}^{BE}$ : Connection weights between input node strain  $\varepsilon_j$  and 1<sup>st</sup> hidden layer node  $B_l$ .
- $w_{ik}^{CC}$ : Connection weights between output node stress  $\sigma_i$  and 2<sup>nd</sup> hidden layer node  $C_k$ .
- $\sigma_i^{NN}$ : The output of the NN. The stress vector.
- $C_k$ : The values of the nodes of the 2<sup>nd</sup> hidden layer.
- $B_l$ : The values of the nodes of the 1<sup>st</sup> hidden layer.
- $NC$ : Number of nodes of the 2<sup>nd</sup> layer.
- $NB$ : Number of nodes of the 1<sup>st</sup> layer.

## 2.6 Neural Networks and Finite Element Analysis

In literature several attempts have been made to include Neural Networks in a Finite Element Analysis. It is important to understand how Neural Networks fit into the Finite Element Analysis, because as a final result the Neural Network Constitutive Models should be functional when part of the analysis. Thus, if they are not able to be incorporated into the final framework then researching them has no practical application. In this section a summary of these attempts will be presented.

### 2.6.1 “Intelligent finite element method” (Javadi & Tan, 2006)

The Neural Network is integrated in a finite element framework. In this methodology the Neural Network will substitute a material model. In this case the Neural Network is trained a priori. The trained network is used specifically in this case to predict the relationship between the stress and strain in the material. This is illustrated in Figure 20. The third validation example in this case is an embankment of Mohr-Coulomb subjected to gravity loading. In this case the soil parameters are obtained from triaxial tests in 5 soil samples. These values are feed into the Neural Network and the Mohr-coulomb material model. However, the actual Neural Network is not attached to the paper so the way that it was trained or how exactly it is implemented in the code is unknown.

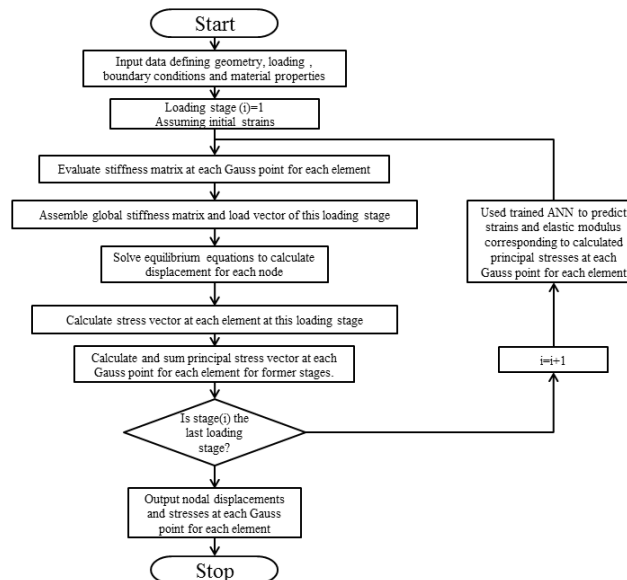


Figure 20 Flow chart of the NeuroFE program (Javadi & Tan, 2006)

### 2.6.2 “A self-learning finite element code” (Shin & Pande, 2000)

In this paper the author aims to create a self-learning finite element code. Here an untrained Neural Network constitutive model (NNCM) is embedded in the finite element code. The finite element code becomes straight forward. The author points out that the NNCM requires a “priming constitutive matrix” to start the neural

network. This matrix usually resembles a linear- elastic matrix of the material. In addition, the NNCM does not require checking for yield computation of gradients of plastic flow rule, updating of yield surface and stress integration algorithms. Therefore, a total stress vector can be simply calculated for a given total strain vector as  $\sigma = \text{NNCM}(\epsilon)$ . In this case the non-linear incremental analysis is undertaken using a two-step staggered scheme. IN the first step (Step 1 Figure 21) the boundary value problem is solved using a “pre-primed” NNCM. The result is a displacement field from which a vector of displacements for the monitored points ( $\delta_c^n$ ) can be assembled. The second step (Step II Figure 21) will produce the stresses and the displacements corresponding to the delinquent displacement vector ( $\delta_d^n$ ) is defined by the equation  $\delta_d^n = \delta_c^n - \delta_m^n$ . These data will be used to train the NNCM. And in that way the finite element code is self-learning. The current stress-strain sets are adjusted in the stress or strain correction scheme. In the strain correction the strain components corresponding to the stress are calculated in Step I and updated in Step II. However, in the stress correction scheme stress, components corresponding to the strain computed in Step I, are updated in Step II. Based on those schemes, when the stress-strain pair become stationary and are within the limits of tolerance the self-learning procedure can be stopped.

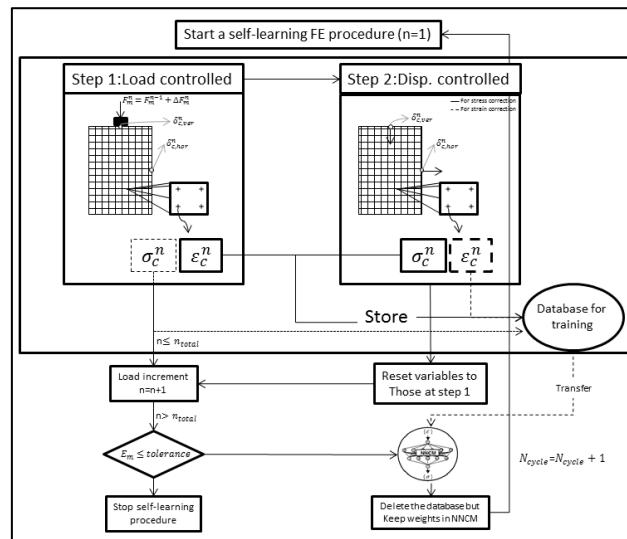


Figure 21 Flow chart of a self-learning FE code

## 2.7 Conclusion of Literature Review

The purpose of the literature review was, first, to understand the computational functions of Neural Networks. After that the purpose of the literature study is to critically review the theory which will be needed for the analysis of the thesis’s results and finally use the results of previous research to identify a promising direction for future research. Through the literature review the typical methodology followed in the process of creating Neural Networks is uncovered. This methodology will be followed throughout the thesis project. In addition, upon reviewing the thesis certain untouched subjects are detected, these subjects are going to be investigated in the thesis project. Most of the reviewed literature fails to deliver a model that is only dependent on the stress-strain input. This will be the main focus of this study which stems from the initial goal of create a more data-driven approach. In addition, an effort to understand why a Neural Network is successful or not usually is not incorporated into the study. Therefore, the thesis will focus on the inner workings that make Neural Networks fail or succeed.

# Chapter 3. First Approach- Generic Neural Network

## 3.1 Introduction

The first attempt in creating a Neural Network material model will be a generic one. In this case the less erroneous Neural Network architectures are achieved through attempting different number of nodes, activation functions and with implementing other callbacks (e.g. dropout, early stopping). As it can be concluded from the literature study little effort is made to link specific components of the material models to specific components of the Neural Network (e.g. activation functions, layers, inputs). Thus, the approach is more generic and the Neural Network can be seen in the end as a black box. However, it has been proved as a successful way to model constitutive soil behaviour by the literature study.

## 3.2 Methodology

To answer the scientific question certain steps need to be defined in the methodology, as it is stated at chapter 2.2. The first step includes creating data for training and testing of the Neural Network, then different network architectures will be tested. The architecture with the smallest errors will be selected as the optimum model architecture (Shahin M. , 2010). The last step includes the validation process where the Network is tested against data that were not included in the training dataset. The last step aims to assess whether the Network was successful in uncovering the relationship between inputs-outputs without overfitting.

### 3.2.1 Creating the dataset

The dataset will be created with the Plaxis (2015) software. To create the data, in the Plaxis software, the SoilTest environment is selected and specifically the General tab. In the General tab the user can define the initial stresses and the strain increments of each phase. The training and testing datasets which are required for the Neural Network's training and validation have to capture the underlining mathematical relationship of each one of the soil material models. Thus, the strain increments have to be selected randomly and the amount of phases has to be large enough to capture the soil behaviour that the Neural Network aims to model.

To facilitate this process a Python script will be created so that Plaxis SoilTest (Figure 22) stresses and strain increments can be imported from a .txt file. Thusly the increments of stress or strain can be selected randomly and a large amount of them can be generated. Finally, the .txt file is formatted in a way that is recognised by the software. The flowchart of this script can be observed in Figure 23.

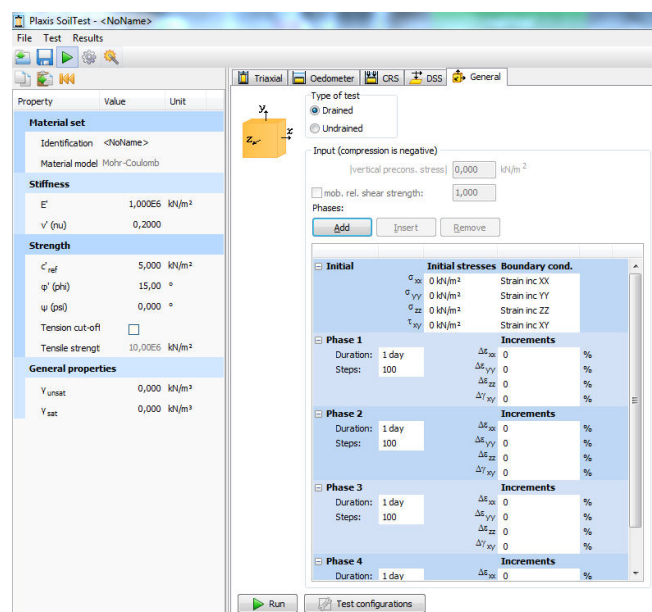


Figure 22 The General tab of the Plaxis SoilTest environment

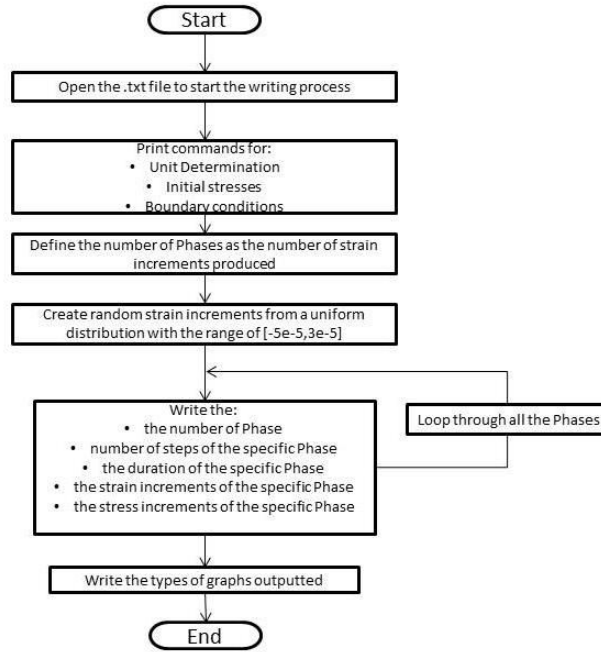


Figure 23 Python script for Plaxis soil test

### 3.2.2 Training Dataset

The dataset consists of 250 strain increments selected randomly from a uniform distribution and ranging between  $-5e-7$  and  $3e-7$ . The strains will be imposed only in the  $yy$  direction. The motivation behind this is that there will be no shear stresses developing in the material and that the stresses in the  $xx$  direction will be equal to the stresses in the  $zz$ . Thus, it is a way of simplifying the relationship the network will attempt to model.

To obtain the training data from Plaxis the material model and parameters have to be defined. In this example a Mohr Coulomb material is selected. The reason being, that it is a relatively simple soil model. The parameters of the soil model are displayed in Table 2. Note that these parameters will remain the same for all the datasets created in this thesis unless stated otherwise.

The final Training dataset can be seen in figures Figure 24, Figure 25, Figure 26 and Figure 27. In Figure 24 the  $p$ - $q$  curve can be observed. In this figure the Mohr-Coulomb failure line can be noticed. In Figure 24 it can be observed that the material will begin with a stress  $p = -100 \text{ kN}/\text{m}^2$  and  $q = 0 \text{ kN}/\text{m}^2$ . As the stress increase the follow the linear elastic relationship until the  $p$ - $q$  combination reaches the failure line. Here the angle by which failure is achieved allows the material to follow the Mohr-Coulomb failure line, resulting in work hardening of the material. In Figure 25 and Figure 26 the actual relationship that the Neural Network will attempt to be fitted can be observed. In the figures the initial loading linear elastic, the linear hardening and the loading unloading parts are shown. Finally, in Figure 27 the strain inputs can be shown.

Table 2 Parameters of the Mohr-Coulomb model

Material Parameters		Value	Unit
Young's Modulus	E	1e6	$\text{kN}/\text{m}^2$
Poisson's Ratio	$\nu$	0.2	-
Strength Cohesion	$c_{\text{ref}}$	5	$\text{kN}/\text{m}^2$
Friction Angle	$\phi$	15	$^\circ$
Dilation Angle	$\psi$	0	$^\circ$

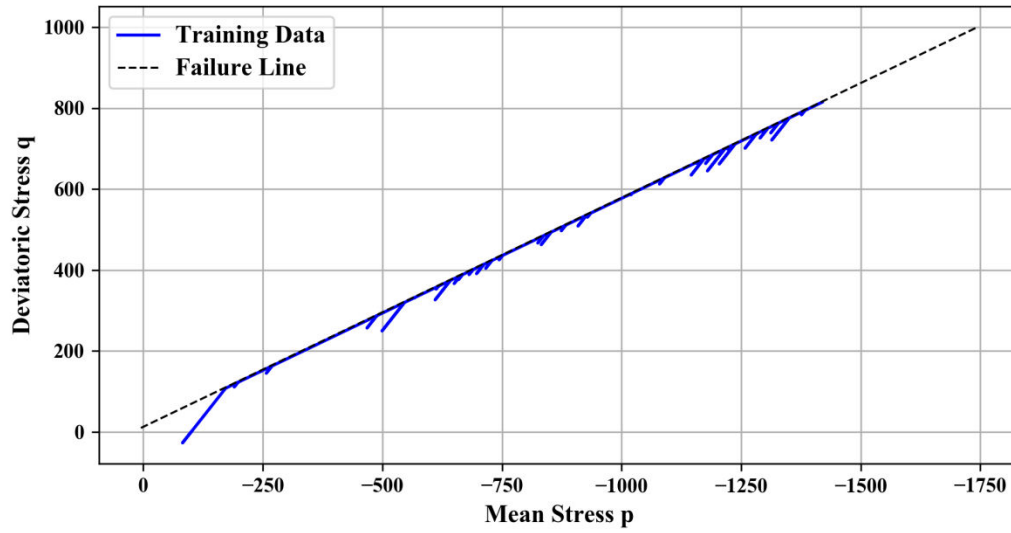


Figure 24 Training dataset p-q curve with failure line

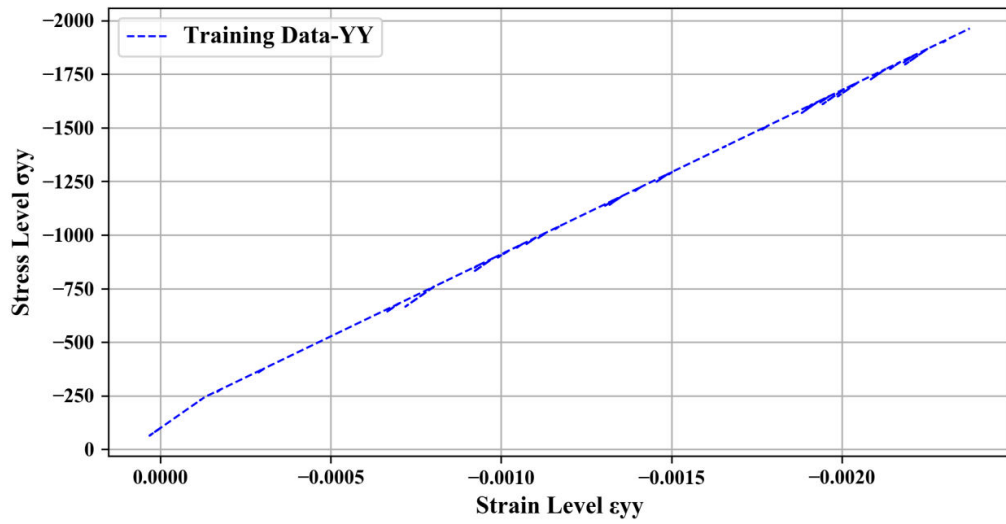


Figure 25 Training Dataset Stress Strain curve

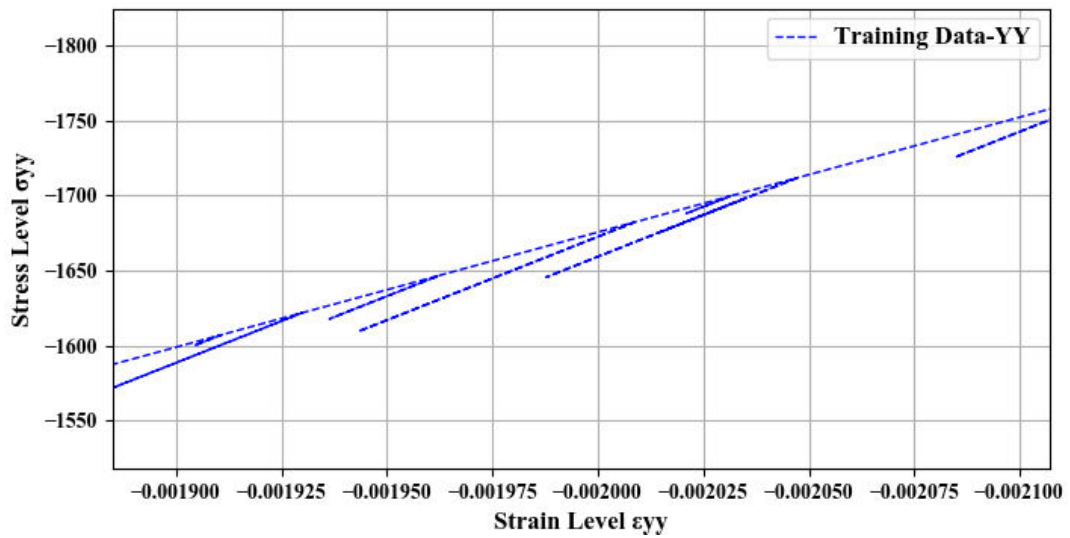


Figure 26 Training Dataset Stress Strain curve Zoomed

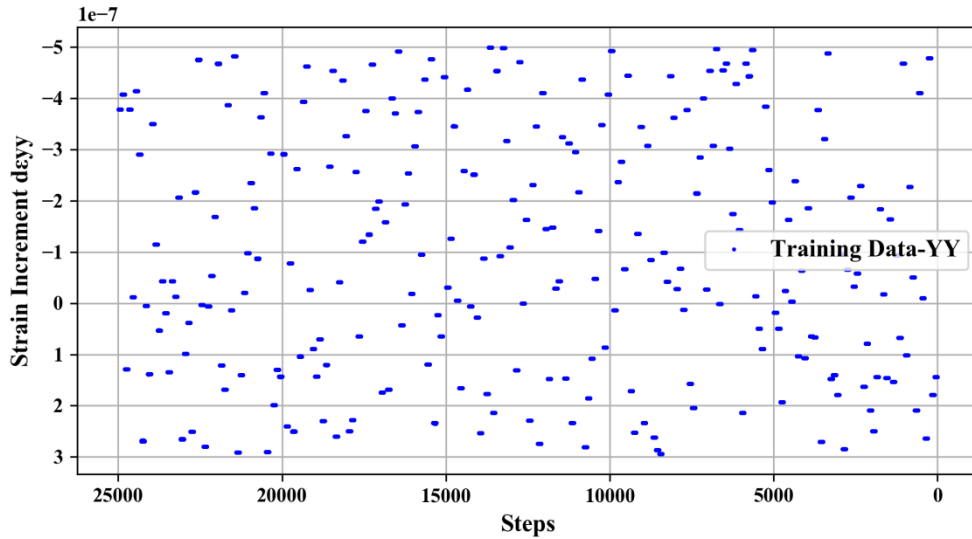


Figure 27 Training Dataset strain increments Vs Steps

### 3.2.3 Creating the Generic Neural Network

To create the Neural Networks a number of programming languages are available with libraries that can be used to facilitated the creation and training of Neural Networks. In this study the programming language Python will be used. The library used is the TensorFlow software library that enables high performance numerical computations. Finally Keras (Chollet, 2015) is used as API, written in Python that is able to run on top of TensorFlow (Abadi, Agarwal, Barham, Brevdo, Chen, & Citro, 2015). The main reason behind these choices is that these libraries and language are relatively easy to use and open sourced. Thus, a lot of material exists online and they are continuously improving and developing due to their popularity among users.

The first step in creating the Neural Network is defining the inputs and the outputs of it. The approach is that the Generic Neural Network should approximate a typical material model. Apart from the material model parameters, that will be kept constant for all the dataset, typical inputs include the strain increments (in this case  $d\epsilon_{yy}$ ) and the stress level (in this case  $\sigma_{yy}$ ).

The neural network will be trained in a way that is depicted in Figure 28(a). However, like most material models the actual way the data will be predicted can be seen in Figure 28(b). The stress level of the previous step is used as an input for the next time step. Thus, for the model to be characterized as successful the results of case (b) have to produce a good fit. The Neural Network prediction model of Figure 28(b) is referred as Feedback prediction in the rest of the document.

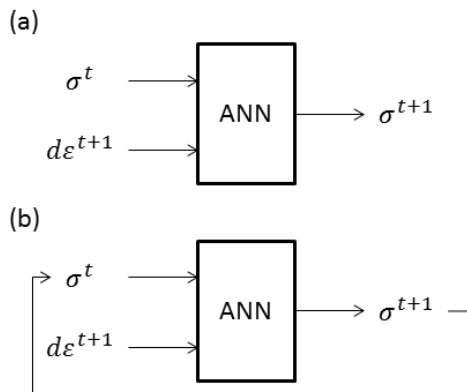


Figure 28 (a) The Neural Network prediction when all the inputs are taken from file (b) The Neural Network when the strain increment is taken from file and the stress state is set with a feedback prediction from the previous step

After defining inputs and outputs the architecture of the network needs to be defined. To find the optimum network architecture different types of them will be tested. The process can be observed in Figure 29. First the activation function of the nodes is selected. Then the number of layers in the network is defined and finally the number of nodes in the layers. The model is trained after that and the relative metrics are calculated to validate



the trained networks performance. These metrics include the mean absolute error, the squared mean error and the coefficient of correlation.

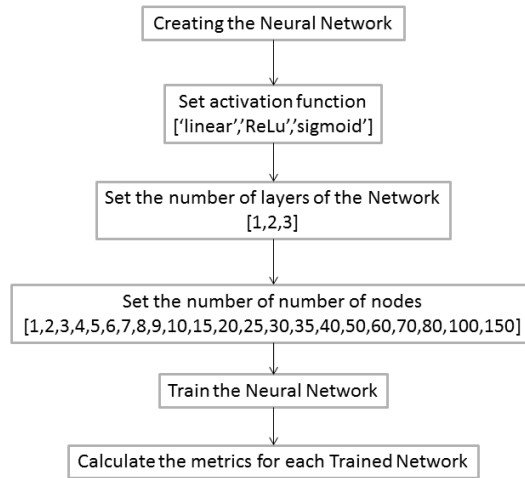


Figure 29 Flowchart of architecture investigation

### 3.3 Results

The results of the generic neural network material model will be presented in this section. In the first part the aim is to determine the optimum Neural Network architecture. In this part the errors computed will be referring to the normal non-feedback prediction. After choosing the optimum Neural Network architecture the next step includes evaluating the results of the network. The results of the Neural Network will be associated with the inner workings of the Neural Network by calculating metrics like Relative Importance and Relative contribution and by displaying the Networks weights.

#### 3.3.1 Network Architecture

As stated in chapter 3.2 the Network's architecture is first defined by the minimum errors and minimum amount of neurons and layers so that the final trained network is obtained efficiently. In Figure 30 to Figure 32 these errors can be observed. Each one of the figures represents a specific activation function that is applied to all of the neurons of the layer. It can be observed that for all the activation functions the errors are smaller when there is only one hidden node layer. The smallest errors can be noted for each of the graphs:

- In Figure 30 (ReLU activation function): 1 hidden layer, 4 nodes
- In Figure 31 (Sigmoid activation function): 1 hidden layer, 110 nodes
- In Figure 32 (Linear activation function): 1 hidden layer, 1 node

After accessing the number of layers of the Network the next step is to determine which activation function performs better. From Figure 33 the optimum activation function is the Linear or the ReLU. For the Linear activation function the optimum network architecture refers to one linear layer consisting of one node and for the ReLU activation function the optimum size is the one layer with 4 nodes, architecture. The final architecture is determined by the generalisation ability of the network. To determine this, a testing dataset will be created. The testing dataset consists of data that are not included in the training dataset. The network architecture that performs the best with the training dataset will be chosen as the optimum one. Table 3 contains the results of the Neural Network with testing dataset the results of the Feedback Prediction are the ones that will determine the optimum architecture. It is concluded that the ReLU architecture has the best generalisation ability.

Table 3 Results of testing dataset for the Linear and ReLU architectures

	ReLU Architecture		Linear Architecture	
	Feedback Prediction	Normal Prediction	Feedback Prediction	Normal Prediction
Mean Absolute Error	0.008832	1.68E-05	0.05767	1.67E-05

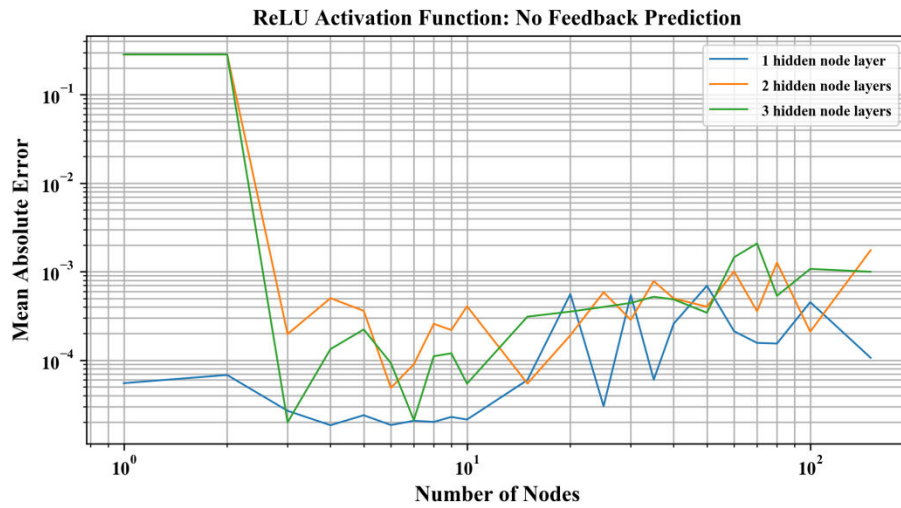


Figure 30 Various architectures and their errors activation function ReLU

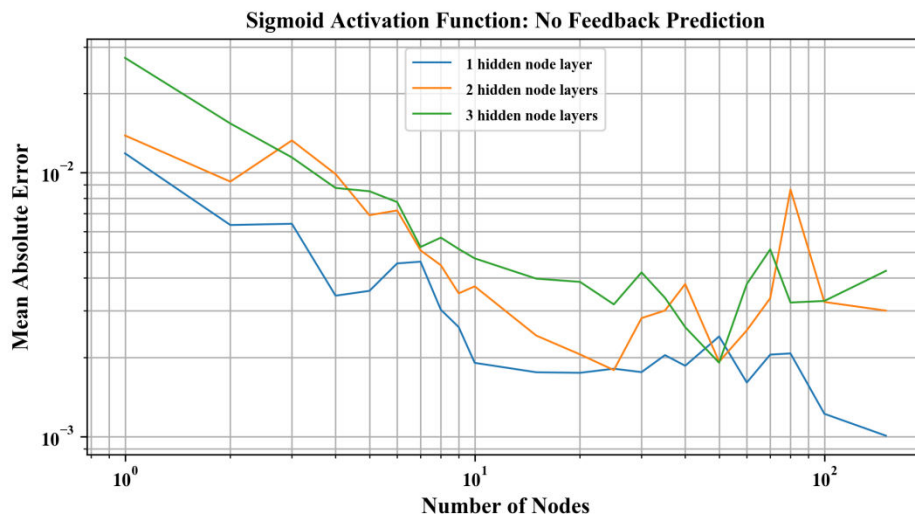


Figure 31 Various architectures and their errors activation function Sigmoid

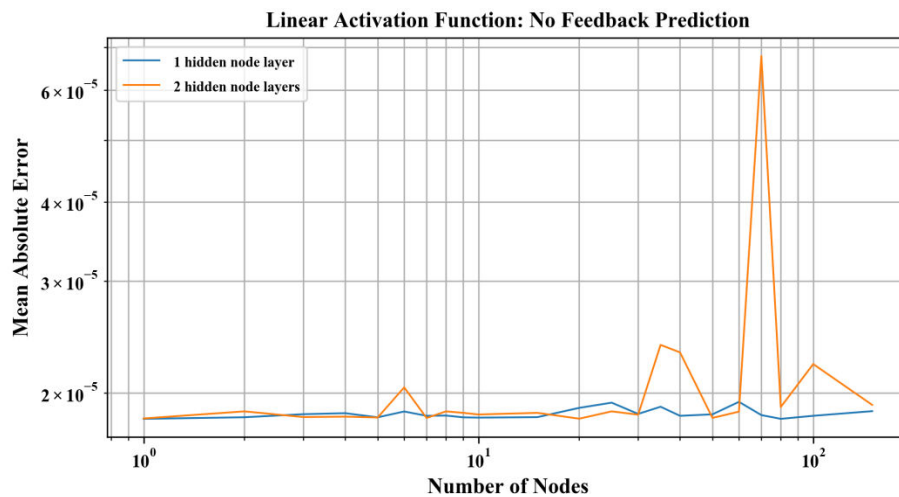


Figure 32 Various architectures and their errors activation function Linear

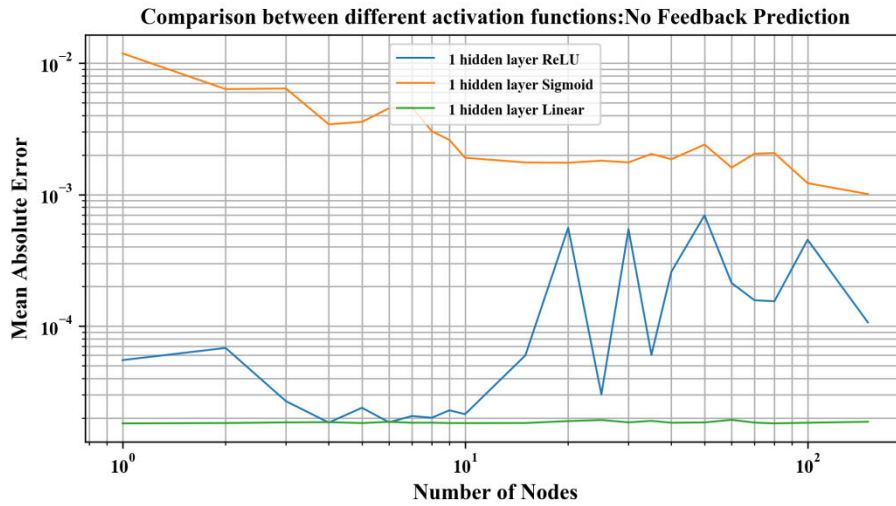


Figure 33 Comparison between different activation functions

### 3.3.2 Results of the Dataset

#### 3.3.2.1 Results without Feedback Prediction

The next step is the validation process of the soil model. The final neural network will have two inputs and one output with a hidden layer of 4 nodes with ReLU activation functions. The trained Neural Network connection weights can be observed in Figure 34. In the figure the strongest connections are represented with a bolder colour and the weakest ones with a lighter colour. Therefore, it is easily observed that the connections related to the input of stress are stronger than the connection of strain. To verify this observation the relative contribution and the relative importance of each output in terms of the weights will be calculated as described in chapter 2.3.3. In Figure 35 the results of this calculation can be seen. First of all, the relative contribution is calculated in a different manner than the relative importance. Thus, their results are in fact different. However, what can be observed in this case is that the contribution of the stress input is definitely bigger than the contribution of the strain increment input. Therefore, the main contribution to the results comes from the stress level. It can be concluded that in this network the stress level changes will have a greater effect in the results.

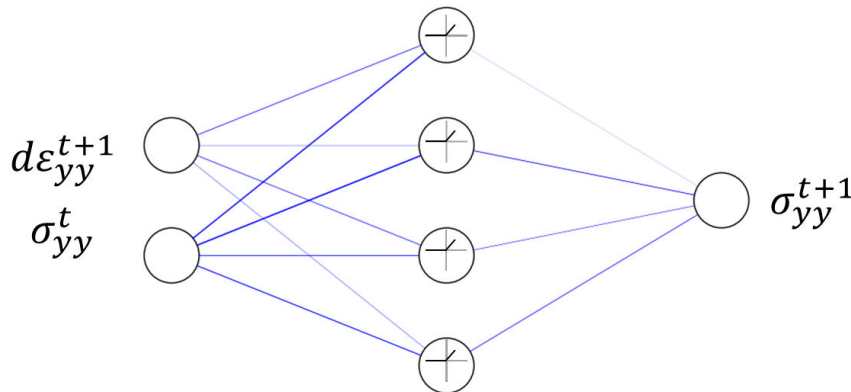


Figure 34 Trained Neural Network bolder connection represent higher weights

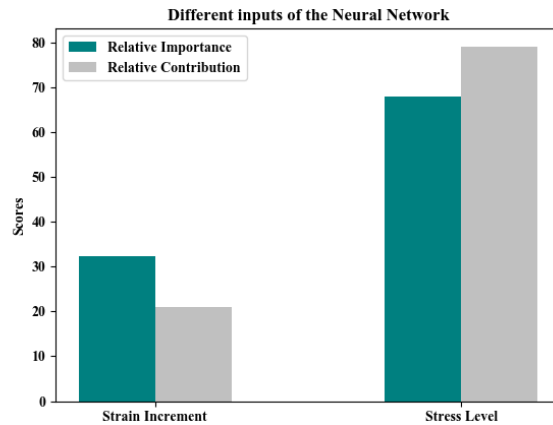


Figure 35 Metrics for weight connections of the Generic Neural Network

The next step in reviewing the results it is validating with a testing dataset. The testing dataset is created in a similar fashion with the training dataset. During training the loss (mean squared error) between actual and predicted results of the network is monitored. The loss of the testing dataset is also calculated after the end of each epoch. The network will uncover the mathematical relationship between inputs and output when both the training and testing losses have converged to a certain value. In Figure 36 the losses for each epoch are converged after epoch 10. Therefore, their general mathematical relationship is “discovered”. The error results can also be seen in Table 4 where various metrics can be represented. As it can be expected the errors of the testing dataset are larger than those of the training. However, the final results (Figure 37, Figure 38) are still acceptable. The correlation coefficient is very high in this case which proves that the target output and the predicted results have a strong correlation with each other.

Table 4 Results of Metrics after training

	Training Dataset	Testing Dataset
Mean Squared Error	5.52E-10	4.34E-10
Mean Absolute Error	1.86E-05	1.68E-05
Coefficient of Correlation	1.000	1.000

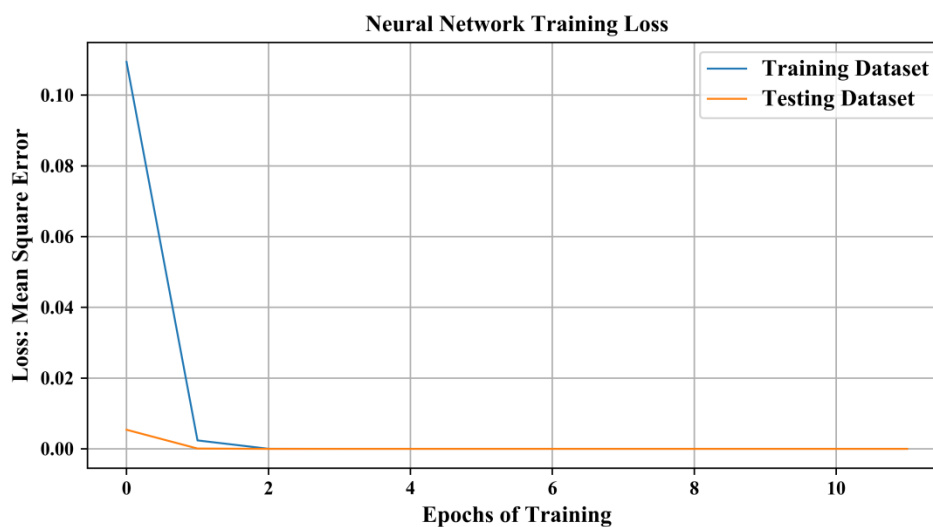


Figure 36 Losses during training for each epoch

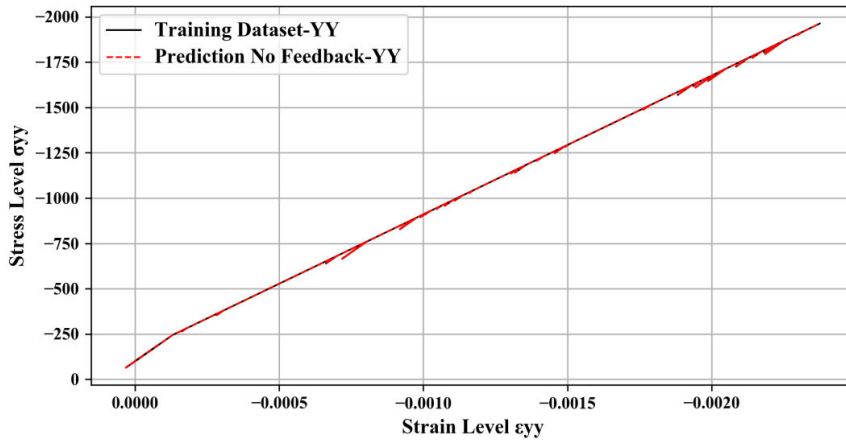


Figure 37 The training dataset results Stress Level Vs Strain Level

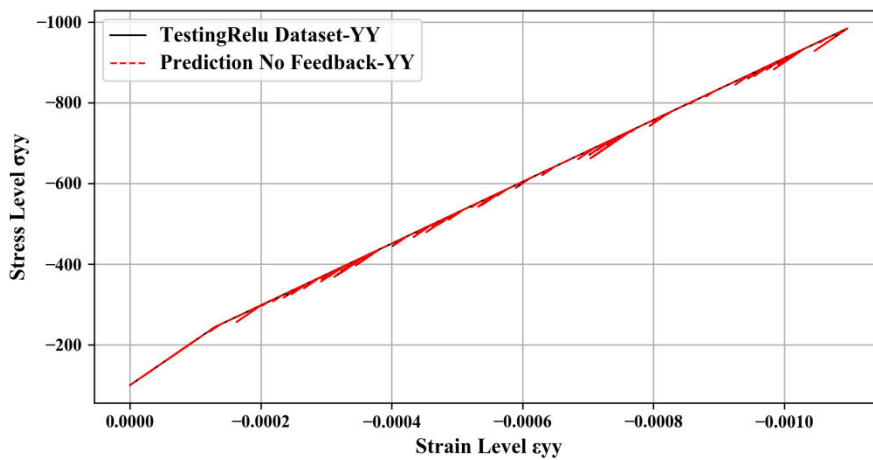


Figure 38 The testing dataset results Stress Level Vs Strain Level

### 3.3.2.2 Results with Feedback Prediction

After the validation of the data predicted without a feedback prediction the next step is to make the prediction sensitive to the previous state. After the training of the model the prediction in this case is made by linking outputs with inputs through a loop (Figure 28(b)). Firstly, the metrics produced after training will be observed in Table 5. The feedback prediction seems to be very erroneous due to the fact that the strain increment's relative contribution is much smaller than the stresses. The result is that with a small error in the stresses has a huge impact in the final output, while changes in the strain have smaller impact on the final output.

Table 5 Results of the Feedback prediction

	No Feedback Prediction	Feedback Prediction
Mean Squared Error	5.52E-10	0.0003
Mean Absolute Error	1.86E-05	0.0144
Coefficient of Correlation	1.000	0.99

To understand why the feedback prediction fails to give the same level of error to the outputs a sensitivity analysis will be conducted to determine the spectrum of input variable contributions in the Neural Network. The sensitivity analysis process involves varying each input variable across an entire range while holding all other input variables constant. In Figure 39 the contribution plot for each of prediction variables is shown. In this case each input is varied across the entire input range while all the other variables are held constant at some chosen percentiles. In the first plot of Figure 39 the results for a varying stress level input can be observed. The stress level input has a linear relationship with the stress level output. In this case the percentile of strain has little effect on the results, because in all of the cases the same linear relationship is followed. On the other hand, the

strain increment input will output almost a straight line. The percentile of stress input seems to be dominating that relationship.

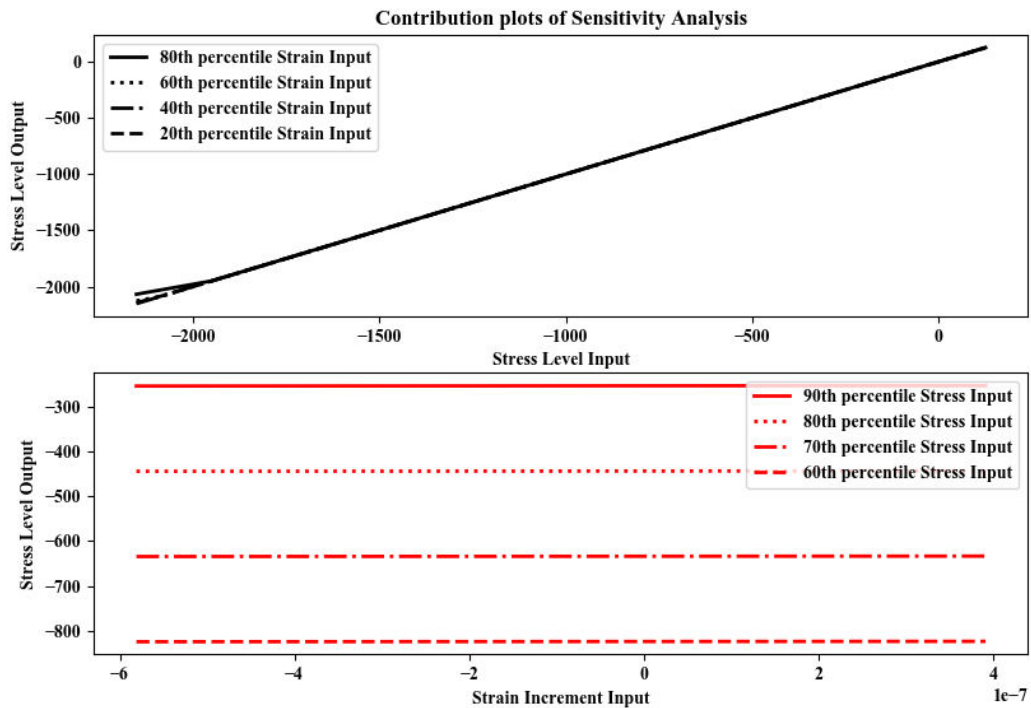


Figure 39 Contribution Plots from the sensitivity analysis illustrating the neural networks response curves to changes in each input with all the variables held constant in different percentiles.

By setting the 80<sup>th</sup> percentile of stress and strain as a constant can be used to determine the range of possible solutions. In Figure 40 the range of solutions can be observed. The strain input ranges between -2000 and  $0kN/m^2$  while the strain increment one between  $-444.4$  to  $-443.6kN/m^2$ .

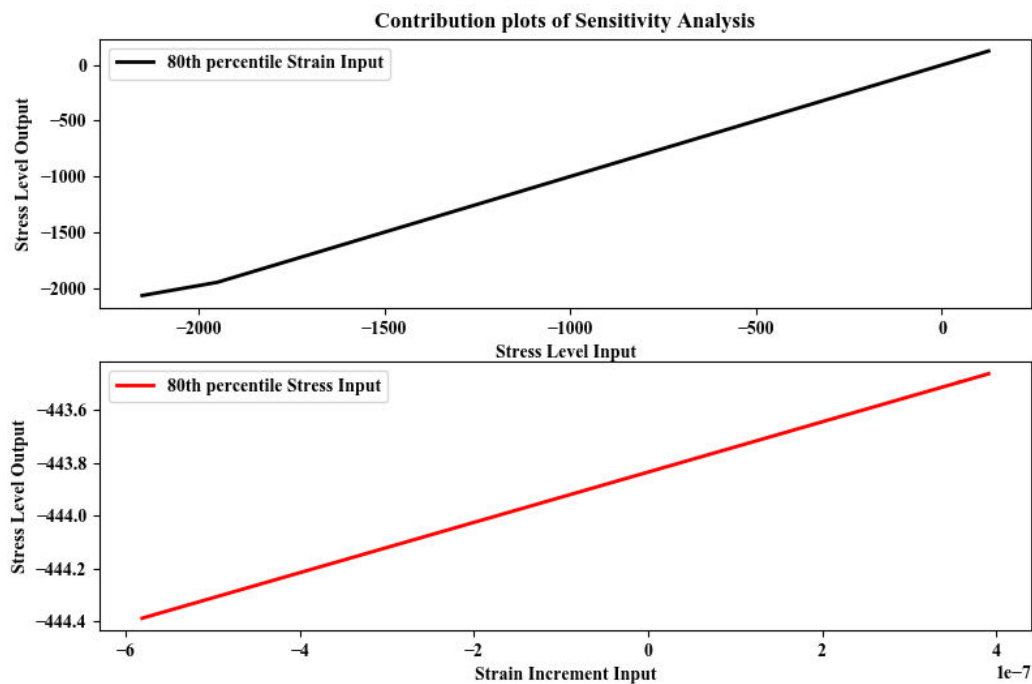


Figure 40 Contribution Plots from the sensitivity analysis illustrating the neural networks response curves to changes in each input with all the variables held constant in the 80th percentile.

In Figure 41 the results of the sensitivity analysis with both of the inputs varying. The x and y axis are the inputs of the Neural Network and the colour represents the output stress level. The inputs are selected randomly from a normal distribution. It can be observed that the strain inputs do not have such a large effect on the output as the stresses do.

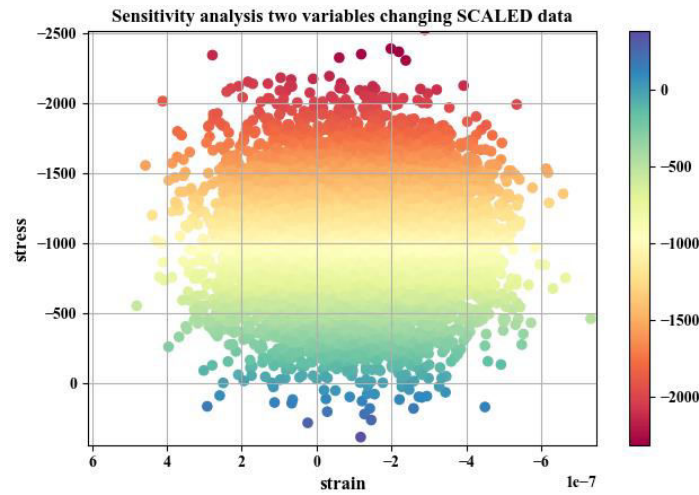


Figure 41 Sensitivity analysis both inputs varying

From the observations made above we can conclude that the output's lack of sensitivity in strain inputs makes the feedback prediction erroneous. In Figure 42 and Figure 43 the actual results of the network can be observed. As it was stated before the miscalculation is attributed to the fact that the stress increment has a greater affect in the final results. However, Figure 43 suggests that the general trends of unloading and reloading are captured in the Neural Network but the feedback prediction causes a drift in the results of the Network. Moreover, in Figure 42 it is observed that the feedback predicts an almost linear relationship between stresses and strains. This suggests that the network does not distinguish between plasticity and elasticity.

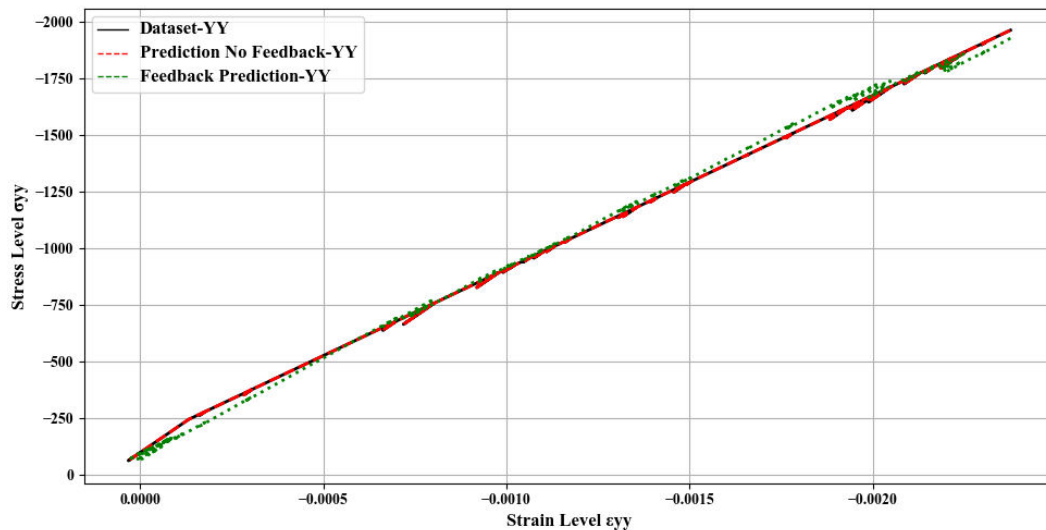


Figure 42 The Training Dataset. The green line representing the feedback prediction

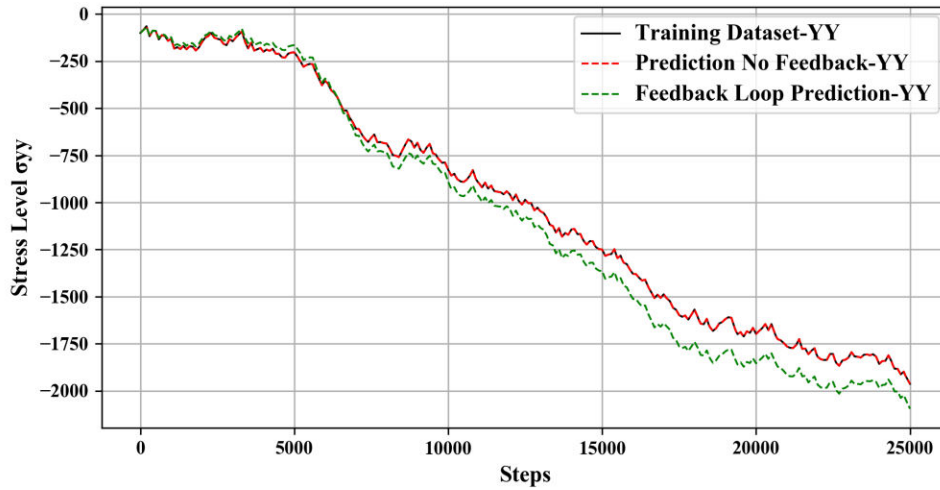


Figure 43 The Training Dataset prediction Vs Steps. The green line representing the feedback prediction

Finally, the inner workings of the network are going to be investigated. The output of each neuron will be calculated to determine how the Network reacts for different inputs. From Figure 42 it can be seen that the feedback prediction approximates a straight line rather than following the hardening yield surface. This implies that in the Network there are not specific patterns corresponding to the elastic and to the plastic behaviour. In Figure 44 it can be seen that those different states are not represented by a different part of the network. The only difference between the Networks is that the stress level input in case (b) is smaller thus the last node of the hidden layer is less active.

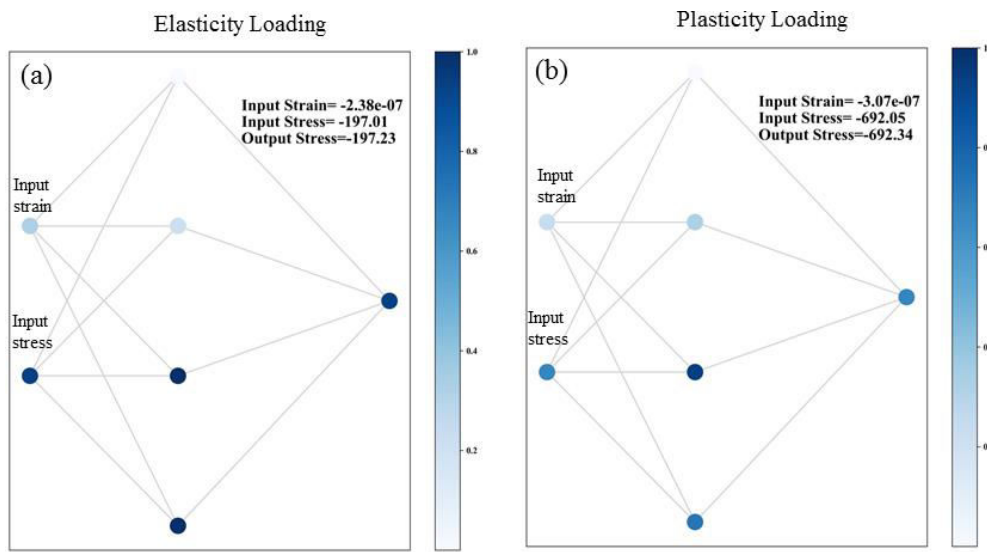


Figure 44 The outputs of each node (a) elasticity (b) plasticity

All of the findings suggest that the network was not trained properly since most of the contribution comes from the stress level input.

### 3.3.3 Improve Neural Network with Feedback Prediction

The feedback prediction has failed to produce an acceptable accuracy for the material model. However, several techniques exist in literature that might be able to resolve the feedback prediction problem. The concept is that if the errors in the training are really low then the feedback prediction errors will be minimised as well.

#### 3.3.3.1 Different inputs and outputs

To improve the feedback of the model the model inputs and outputs have to be reconsidered. One way to approach this problem is to find already successful models in literature and draw inspiration from them in terms of inputs and outputs. The new input array will be consisted of  $d\varepsilon_{yy}^{t+1}$ ,  $\varepsilon_{yy}^t$ ,  $\sigma_{yy}^t$  and the output is changed to  $d\sigma_{yy}^{t+1}$ . Thus, the strain level is added as an input and the stress increment in the next step is the new output.



(Sidarta & Ghaboussi, 1998) To find a suitable architecture a similar approach will be implemented as it was in chapter 3.3.1. This architecture might be more successful with the feedback prediction because the stress level input will not have such a large correlation with the stress output. Thus, the contribution of the stress input will be minimised.

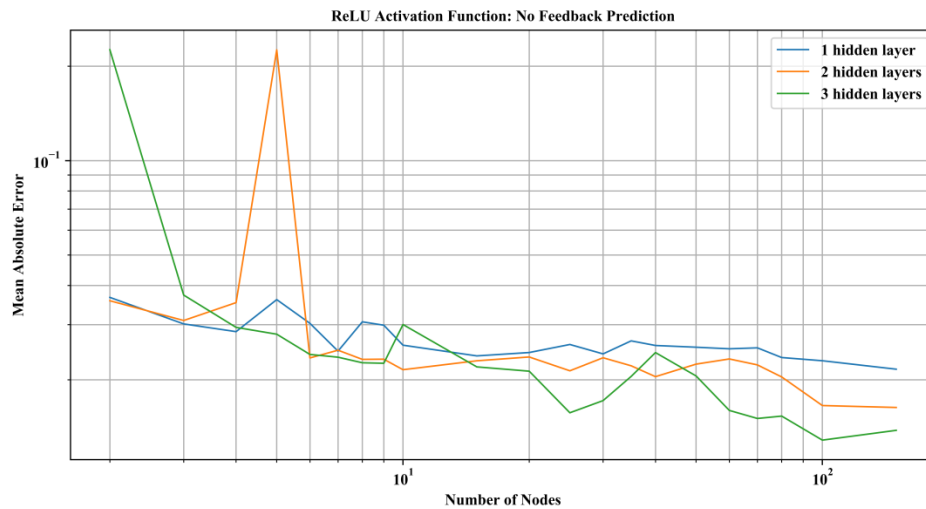


Figure 45 The new inputs and outputs Training Dataset Errors

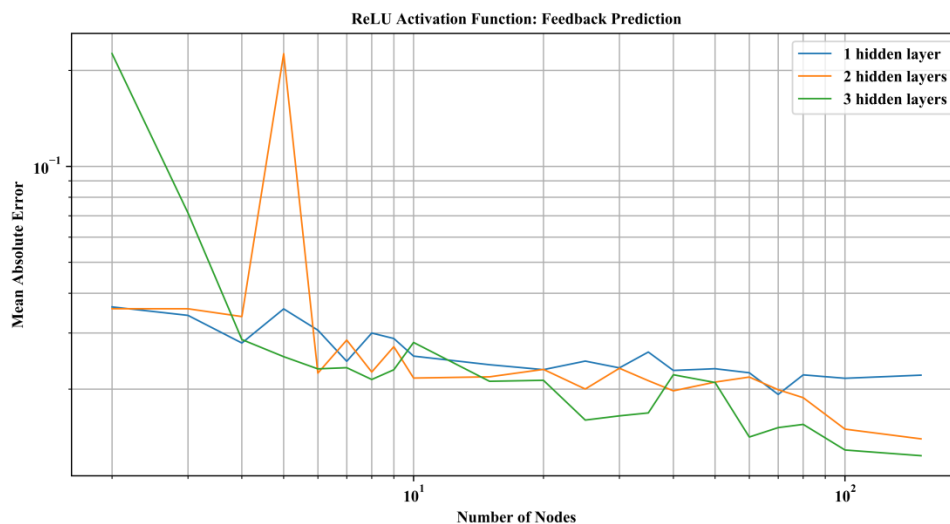


Figure 46 The new inputs and outputs Training Dataset Errors Feedback prediction

From Figure 45 the optimum architecture of the Neural Network consist of three layers and 150 nodes. However, these errors refer to a non-feedback prediction. Therefore, to determine the optimum architecture the errors of the feedback prediction need to be defined. In Figure 46 the minimum error in the Feedback prediction is achieved with architecture of 3 layers and 150 neurons. Figure 45 and Figure 46 have small differences with each other in terms of errors. This implies that the issues of chapter 3.3.2.2 were resolved.

However, when the results are observed in Figure 47, it is concluded that this network does not perform more accurately from the network in chapter 3.3.2.2. From the figure it can be seen that the feedback prediction will perform better than the normal prediction. However, zooming into the graph it is easy to observe that the prediction is still not accurate. Finally, the relative importance of each input can be calculated. In Figure 49 the Relative contribution of the inputs reveals a balanced outcome with the strain level contributing more to the output.

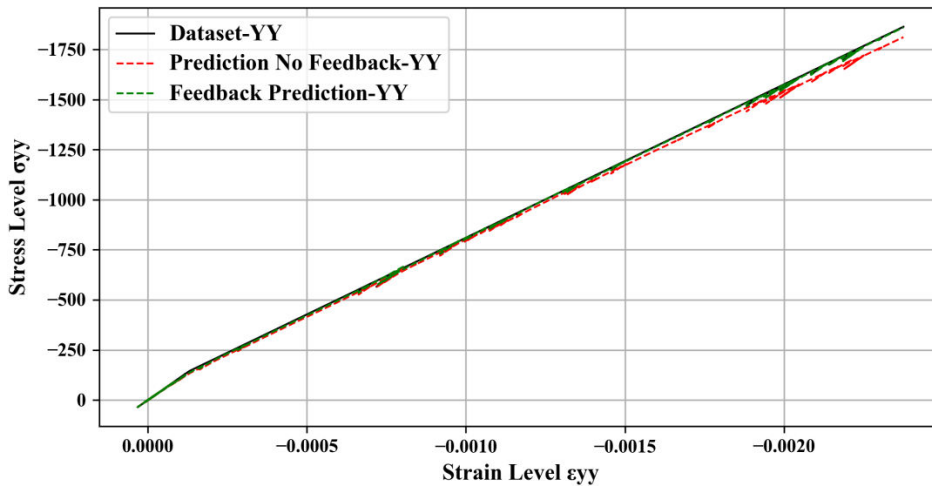


Figure 47 Training dataset results New inputs and outputs

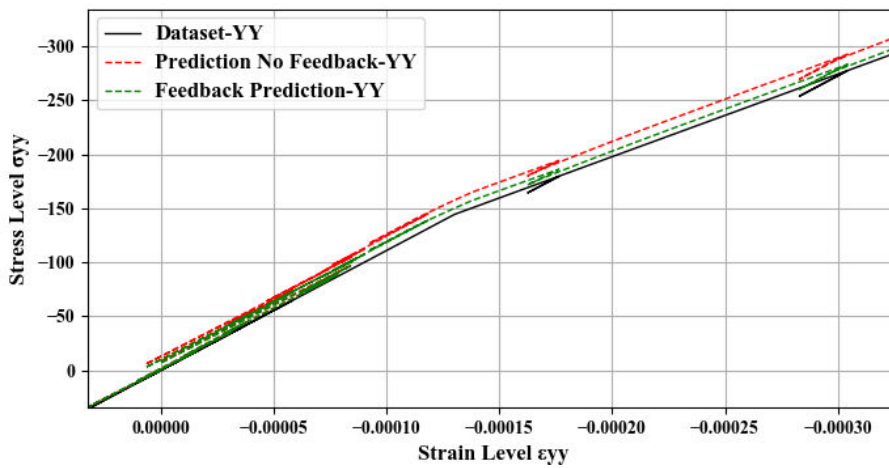


Figure 48 Zoomed training dataset results New inputs and outputs

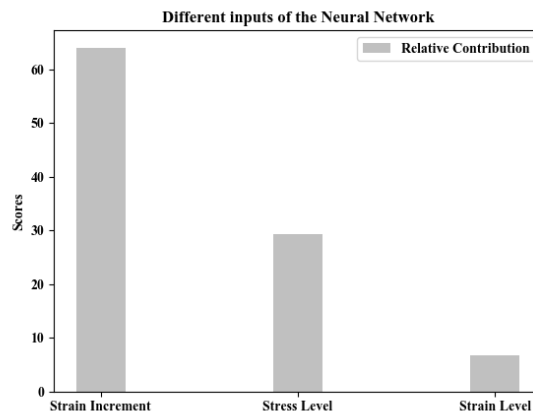


Figure 49 Relative Contribution of each input- New inputs and Outputs

A sensitivity analysis is also performed to further evaluate the Neural Network. The sensitivity analysis is implemented in a similar fashion as in chapter 3.3.2.2. The results can be observed in Figure 50. In Figure 50 the results of the Network when only one input is varying can be seen. From the first plot of the figure the results of a varying stress level are observed. The relationship in this case is a decreasing response. Thus, a decrease in the stress level input will lead in an increase in the stress increments no matter the percentile input chosen. However, percentile inputs seem to have an effect on the range of responses. For example a small

percentile input gives the ability to the output to vary significantly. The effect of the stress level is bigger than expected. Theoretically, the stress level should be quite irrelevant to the stress increment output since the only function of the stress input is to distinguish plastic from elastic behaviour. Moving on the second plot, where the strain increment inputs vary, there is a linear relationship between the strain input and the stress output. It is also noticed that the negative parts of the Inputs correspond to negative parts of the output. However, the percentile inputs influence the stress increment output. Thus, high levels of stress level and strain level inputs will result in non-zero stress increment output which is not correct from a constitutive model point of view. Finally, the last plot shows the relationship between varying strain level input. It can be observed that the range of solutions consists mostly of negative stress increment outputs. The response of the Network in this case is classified as increasing because when the strain level increases the stress level increases as well. In addition, it is observed that the percentile inputs in this case do not have a clear influence on the output.

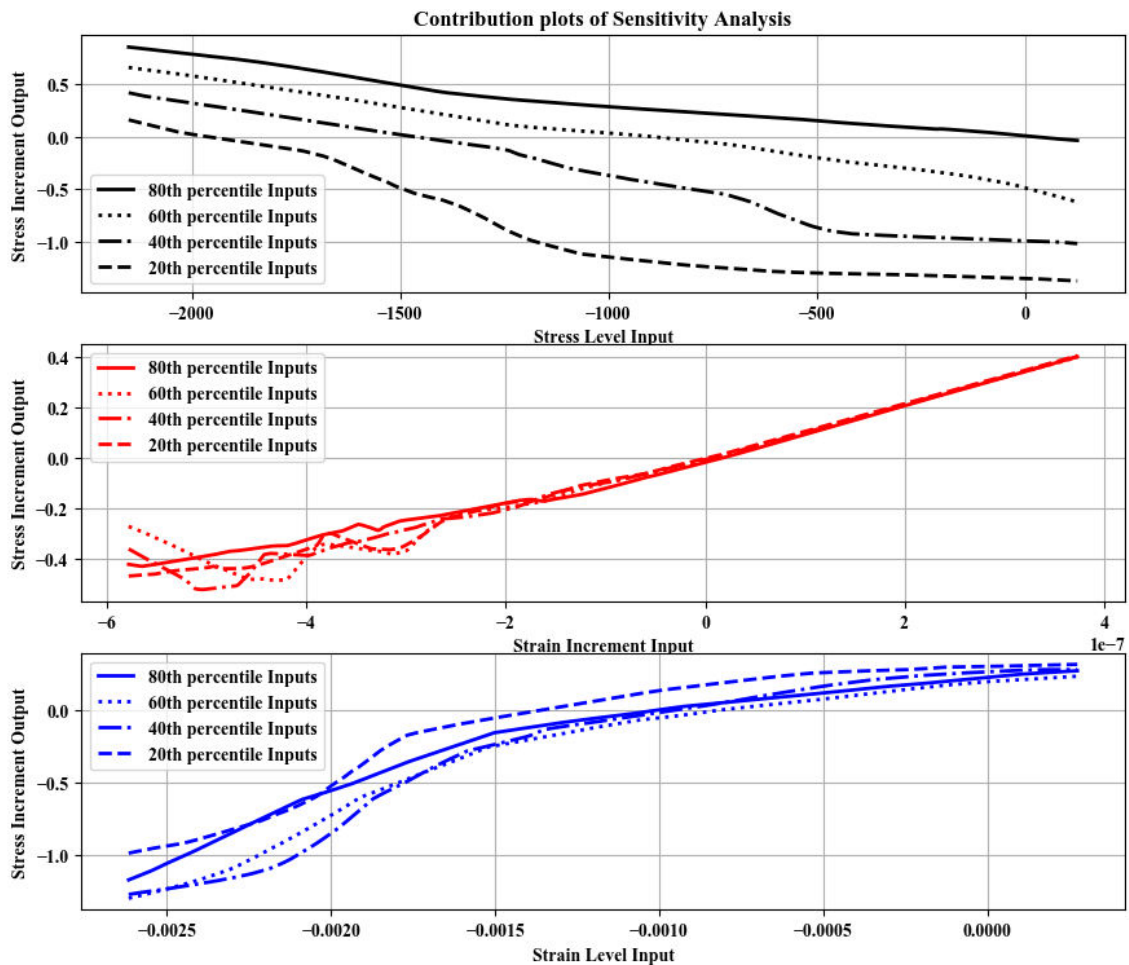


Figure 50 Contribution Plots from the sensitivity analysis illustrating the neural networks response curves to changes in each input with all the variables held constant in different percentiles.

### 3.3.3.2 Dropout

Dropout is described in chapter 2.3.4.2. Here it is stated that by implementing the Dropout method one can check multiple network architectures efficiently. Thus, instead of increasing node and layer size, a large amount of layer and nodes is chosen and then the Dropout method is applied to them. The initial architecture in this case is chosen as 2 layers with 150 nodes per layer this architecture is chosen because it possesses a large number of connections and thus more architecture will be tested through the dropout method. The rate of the dropout will define how many of the connections between nodes to drop. In this case the rate will be 0.5 so 50% of the connections will be deactivated randomly during training. However, in this case the Dropout technique did not help with the generalization of results and thus the feedback prediction is still erroneous. In Table 6 the results of the equivalent network without any dropout is presented. Although the errors are diminished they are not in the order of magnitude that it is required.

Table 6 Results of the Dropout Neural Network compared to a Full Connected one

	Dense Connection	Dropout
Mean Absolute Error	0.0003	0.293
Mean Squared Error	0.0144	0.11
Coefficient of Correlation	0.99	0.009

### 3.4 Conclusions

In this chapter a generic Neural Network Constitutive Model is created. The first attempt is to create a Neural Network consisting only of stress and strain inputs, in that way the Constitutive Model of the Neural Network does not require soil parameters and thus represents a data driven approach. Several architectures are attempted and the most efficient and accurate one is chosen. That architecture consists of one hidden layer of 4 neurons with a ReLU activation function. Although the network performed satisfactory with a normal prediction the feedback prediction proved to be quite erroneous. This is attributed to the fact that the contribution of the stress inputs is significantly larger than the strain inputs. Therefore, the contribution of the stress input exceeds the contribution of the strain input which will lead to a false estimation of the stress output during feedback. To resolve this issue several methods are implemented. Different inputs and outputs are tested as well as the regularisation method of Dropout. By altering the inputs and the outputs of the network the network was successful in producing the same results with a normal and a feedback prediction. That finding reinforces the idea that the first Neural Network was not successful because of the varying contributions of the inputs.

In all of the methods tested the error of the feedback prediction is too large for the Network to be successful. Thus, the approach of creating a generic model has to be reassessed. As it can be implied from the approach of this chapter the Neural Network was created by not taking into account how constitutive models work in detail. That led to a Neural Network which overestimated the effect of one of the inputs. However, the next step would be to link soil behaviour with Neural Network architectures. Soil constitutive models usually consist of different components (e.g. spring element, dashpot element) that represent different soil behaviours (e.g. elasticity, creep). When these components are linked together the full stress-strain relationship of the constitutive model can be realised. In the same way the Neural Network constitutive model can be rearranged into components that are then connected, to represent the full stress-strain relationship.

Nonetheless, there are more possible remedies for the problem of feedback prediction. One of those is adding history stress points as inputs. In this way the stress level can correspond to a certain history input and thus the feedback prediction can be defined more accurately (Ghaboussi, Pecknold, & Zhang, 1998). Another possibility is to train the model by using a recurrent neural network. In this case the training is achieved with feedback and the network might be more successful in uncovering the stress – strain relationship (Zhu, Zaman, & Anderson, 1998).

# Chapter 4. Second Approach- Component Based Modelling

## 4.1 Introduction

The generic Neural Network material model created in Chapter 3 failed to capture the stress strain relationship of the soil with the feedback prediction. To ensure a better prediction the “problem” has to be redefined. In this case the “problem”, so whether Neural Network can model constitutive soil model is going to be assessed from another perspective. Typical components of constitutive models will be identified. These components will be modelled by using Neural Networks. The components will be linked together to create the full stress-strain behaviour. In this chapter the material models created in this way are the linear-elastic model, the linear-elastic perfectly-plastic and finally the linear elastic work-hardening. First, the choice of architecture for each Neural Network is explained in chapter 4.2. In chapter 4.3 the results of the training and the validation of each Neural Network model are explained. Finally, conclusions for each model are made in chapter 4.6.

## 4.2 Methodology

The first part of the methodology explains how components construct material models when they are linked together. Furthermore, each of the constitutive soil models used in the Neural Network training process will be explained in detail, so that the dataset corresponding to their behaviour can be created successfully. In the last part of the methodology the idea behind each of the component based Neural Networks is described.

### 4.2.1 Typical components of material models

Conceptually all material models can be constructed by using three simple components (Figure 51 a, b and c) shown in Figure 51 the spring, the dashpot<sup>3</sup> and the slider. The linear elastic component (a) is modelled as a reversible spring; the creep is represented as a dashpot element and the plasticity with a slider plastic resistance element. From these components more complex soil behaviours can be formulated, such as plasticity or viscoelasticity. Assembly of the components has been proven as a logical way of generating the soil behaviour pattern. One of main scientific questions is if Neural Networks can behave as successful as material models. To answer this question one has to consider if Neural Networks are able to capture the typical components of material models. Thus, if different components of the soil material models can be represented by different parts of a Neural Network then the hypothesis, that Neural Networks can model typical soil behaviour, is proven.

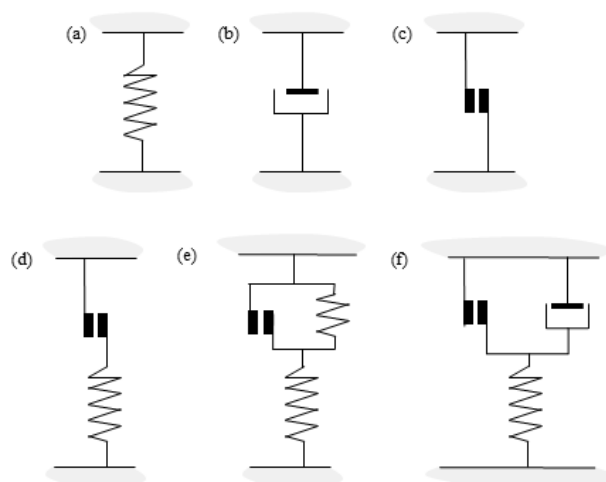


Figure 51 Basic components for material models. (a) Spring-reversible linear/nonlinear elasticity. (b) Dashpot-linear/nonlinear creep. (c) Slider-plastic resistance (strain dependant). (d) Elasto-plastic model (e) Work-Hardening Plasticity (f) Elastic viscoplastic assembly. (Zienkiewicz, 1985)

<sup>3</sup> Dashpots are simple pistons combined – usually – with a hydraulic fluid. Examples of objects that include dashpots are car shocks, bike shocks, and are found on some doors.

#### 4.2.1.1 Linear-Elastic

The linear elastic model is the equivalent of Hooke's law. In this case stresses and strains are connected through a linear relationship. In this case strains are completely reversible since the only state of the material is elastic. The relationship can be expressed in the form of:

$$\bar{\sigma}' = \bar{M}\bar{\epsilon} \quad (18)$$

Where M is the stiffness matrix. Expressed from Hooke's law the relationship becomes:

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{zx} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{yz} \\ \epsilon_{zx} \\ \epsilon_{xy} \end{bmatrix} \quad (19)$$

E': Young's Modulus

$\nu$ : Poisson's Ratio

Thus, the relationship between stress and strain is represented by a straight line usually through the origin. (Figure 52) The behaviour of the linear elastic material is further exemplified by the stress time and strain time curves. In linear elastic behaviour when a soil body is subjected to a constant strain  $\epsilon_0$ , the stress will reach a certain constant stress level  $\sigma_0$ . When the strain is removed at time T the stress will return to zero. (Figure 53)

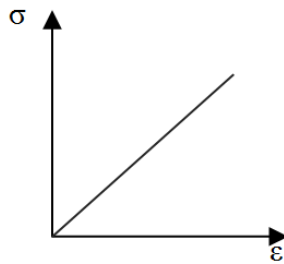


Figure 52 The stress strain relationship of linear elastic

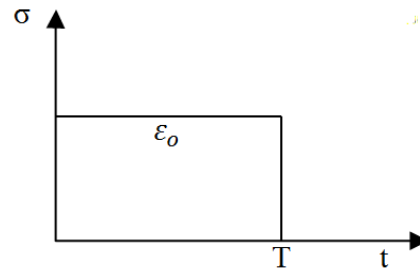


Figure 53 Linear-Elastic Stress-time curve

#### 4.2.1.2 Linear-Elastic Perfectly-Plastic

A plastic material is defined by the fact that it does not undergo plastic deformation until certain yield stress has been exceeded. In addition, if a material is classified as perfectly plastic (Figure 51c) no elasticity will occur during loading. The stress-strain relationship is shown in Figure 54, under the influence of small stresses, no deformation occurs; when the stress is increased the material will reach the yield stress of  $\sigma_0$ . Until the stress is removed the material will remain at this constant stress level and the strain level accumulated by this process is retained.

However, most soils are closer to elastic-plastic behaviour. Here the slider-plastic resistance can be represented as a block of solid material on a flat horizontal surface. Thus, when a force is applied to it the block will not move until the force exceeds the friction between the block and the flat surface. The ideal linear elastic perfectly plastic model is shown in Figure 51d.

There are two main components Mohr-Coulomb model. The elastic part which is described by Hooke's law and the Plastic part that is based on the Mohr-Coulomb failure criterion formulated in a non-associated plasticity framework.

The plasticity results into non-reversible strains. The yield function,  $f$ , determines whether or not plasticity has occurred. This is represented as a surface in stress space. For stress states represented by points in the yields surface, the behaviour is purely elastic and all strains are reversible.

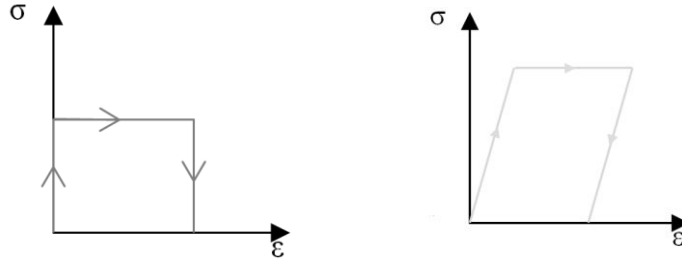


Figure 54 The perfect plastic model    Figure 55 The linear elastic perfectly plastic model

#### 4.2.1.3 Linear-Elastic Work Hardening Model

In this model the elastic and plastic curves are assumed linear. If we consider a loading-unloading case the soil element will be loaded until the yield stress level. If the loading continues the soil element will go into the plastic region and thus will follow a plastic linear relationship. If the soil element is unloaded after some deformation has taken place and then reloaded, the reloading portion will approximate a straight line of slope  $E$  until the highest previously attained stress (the stress of the soil element before the loading began). Reloading will follow the virgin curve. The highest stress before the unloading will be the new yield stress. The material is, thus, considered as been strengthened or hardened by this plastic deformation. Therefore, work-hardening or strain-hardening is the increase of stress with plastic deformation. The **Ramberg Osgood** formula can describe this behaviour as:

$$\varepsilon = \frac{\sigma}{E} + \alpha \frac{\sigma_R}{E} \left( \frac{\sigma}{\sigma_R} \right)^m \quad (20)$$

$\alpha, m$ : dimensionless constraints

$\sigma_R$ : reference stress

If  $m$  is very large the plastic strains will remain small until the stress approaches the yield stress and then increases rapidly when stress exceeds yield stress. In the limit, as  $m$  becomes infinite, the strain is zero when  $\sigma < \sigma_R$ , and is intermediate when  $\sigma = \sigma_R$ , while  $\sigma > \sigma_R$  would produce an infinite plastic strain and therefore impossible. This limiting case accordingly describes a perfectly plastic solid with yield stress  $\sigma_R$ . Finally, the stress-strain relations can be expressed as in equation and Figure 56(c):

$$\sigma = \begin{cases} E\varepsilon & \text{for } \sigma \leq \sigma_R \\ \sigma_R + E_t \left( \varepsilon - \frac{\sigma_R}{E} \right) & \text{for } \sigma > \sigma_R \end{cases} \quad (21)$$

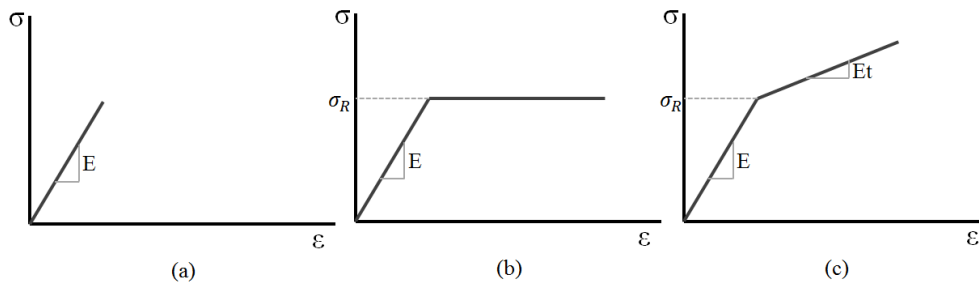


Figure 56 The Material Models

From a continuum mechanics point of view, represented in Figure 51(e), the components corresponding to plasticity are a spring and a friction component. To produce the final relationship seen in equation 21 these components are connected in a parallel manner. The elasticity is represented by a linear component attached in-series to the system.

#### 4.2.2 Create the Training Datasets

The datasets are created in a similar fashion as in chapter 3.2.1 in the Plaxis software from the SoilTest function and the General tab. Each one of the datasets, however, will be created differently. Furthermore, the material parameters used in all of these cases are posted in Table 2.

#### 4.2.2.1 Linear Elastic

##### 4.2.2.1.1 One Direction of Strain Input Dataset

The linear elastic model is created in Plaxis SoilTest function General tab. The strain increments (thus the inputs to the Neural Network) are only considered in one direction. These increments are randomly selected from a uniform distribution between  $-5e-7$  and  $3e-7$ . In Figure 57 the linear relationship between stresses and strains is observed for all of the stress output directions. Directions XX and YY will have equal stress outputs since they are governed by the same linear relationship. In Figure 58 the stress level with relation to the steps is presented.

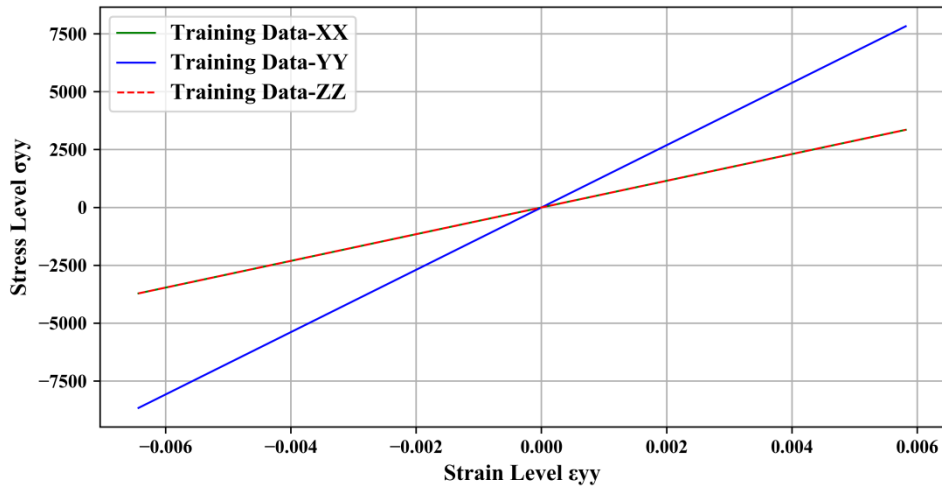


Figure 57 The linear relationship between stress and strain

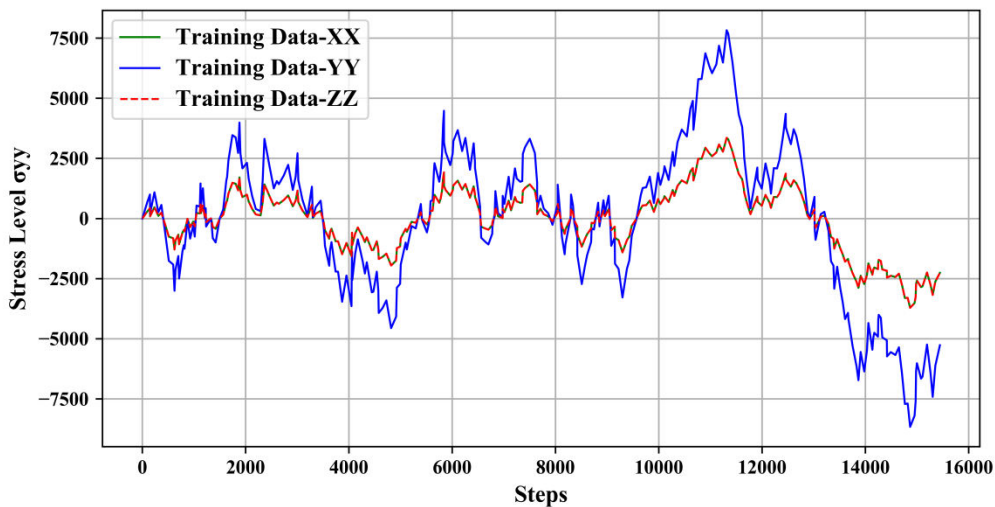


Figure 58 The stress level at each of the steps

##### 4.2.2.1.2 Three Directions of Strains Inputs Dataset

The next dataset will be used for the Neural Network modelling three dimensions of inputs and outputs. Thus, the inputs in the linear elastic material model consist of three directions of strain increments. Each of the directions will consist of 200 increments that are selected randomly from a uniform distribution with limits  $-2e-05$  and  $2e-05$  (Figure 59). Imposing three random increments at a time will result in the final stresses as they are presented in Figure 60. Each of the directions will follow a different stress path since the linear relationships governing them are different from each other.



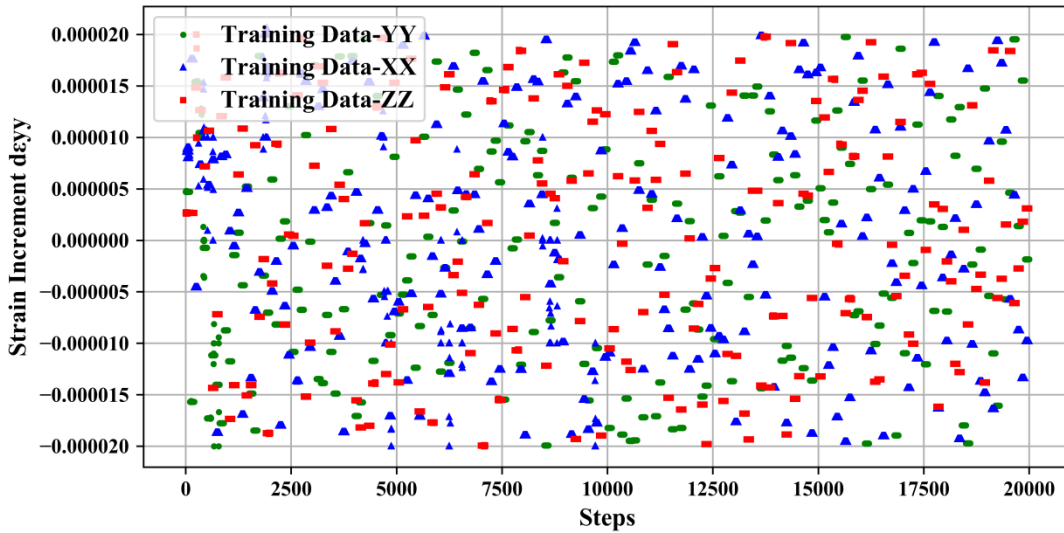


Figure 59 Strain increments for the 3D elastic dataset

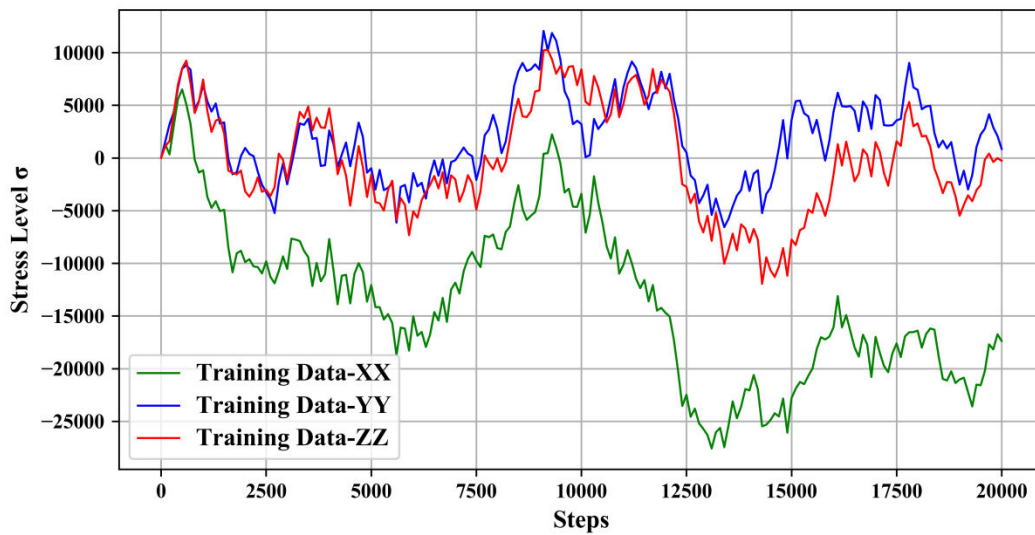


Figure 60 Stress Level of the 3 directional elastic dataset

#### 4.2.2.2 Linear Elastic Perfectly Plastic

The next dataset created is the linear-elastic perfectly-plastic. In this case the soil element will reach perfect plasticity. The aim of this dataset is to reach a constant yield stress level where no more stresses are allowed to accumulate.

In this case the generation of strain increments will come from a random uniform distribution. However, in this case the test is regarded as a “triaxial” test as it is defined in the PLAXIS software. Thus, one direction will be the dominant one (direction y) with strain increments that will be randomly selected from a uniform distribution. The two remaining directions will have an increment of half of the y directions in the opposite direction. The result can be observed in the p-q curve (Figure 63) in this case the mean stress will remain constantly the same and the deviatoric stress will vary from 0 to 67,22kPa depending on whether the soil element is loading or unloading. In Figure 61 the stress strain relationship can be observed. The diagram has two parts an elastic loading-unloading part and a plastic constant stress part.

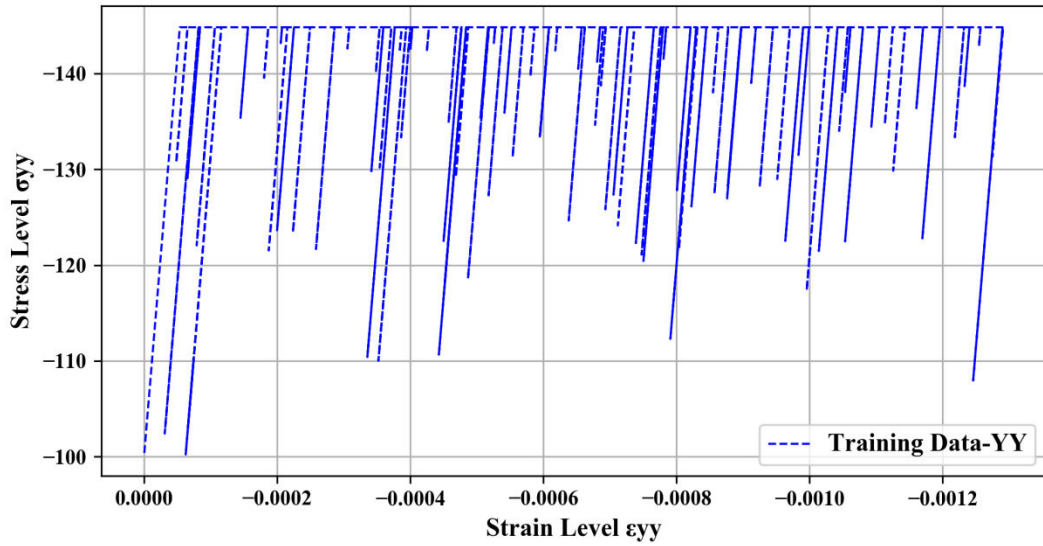


Figure 61 The training dataset's stress to strain relationship

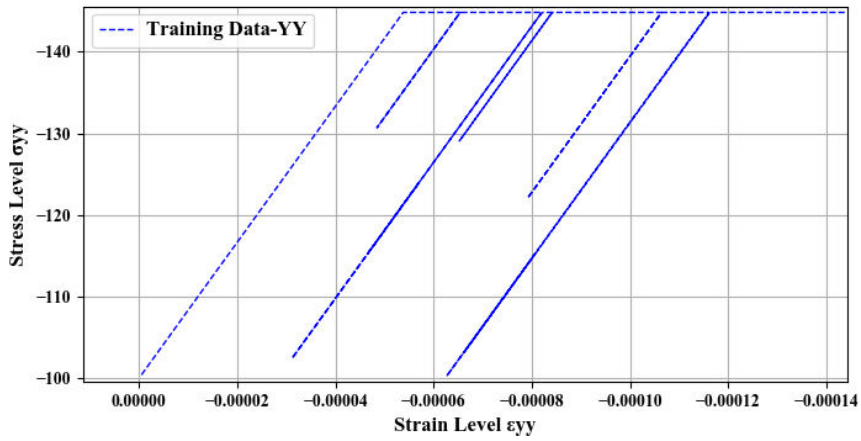


Figure 62 The training dataset's stress to strain relationship Zoomed

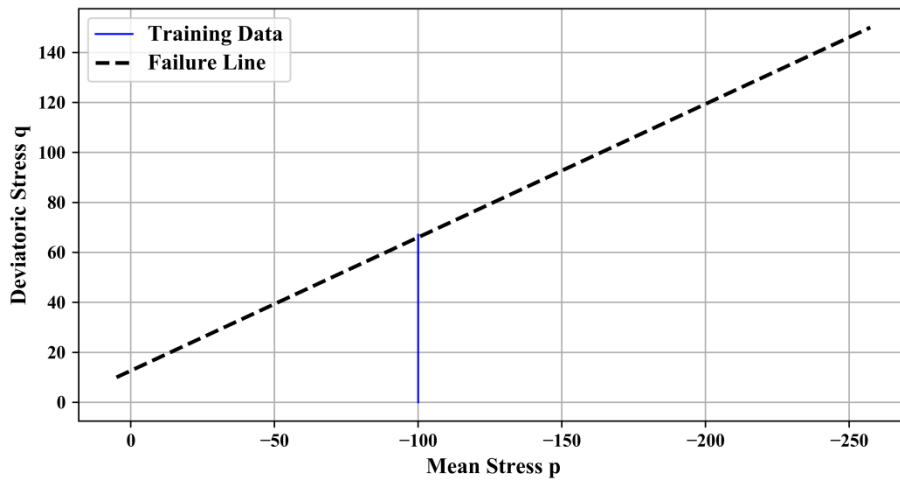


Figure 63  $p$ - $q$  curve for the linear elastic perfectly plastic dataset. The black line represents the failure line of the material

### 4.2.2.3 Linear Elastic Work Hardening Model

This dataset will be created in the same manner as the linear elastic. The strain increments are randomly selected from a uniform distribution. The strains are imposed only in the y direction. In the p-q curve the mean stress reaches the yield surface in a non-perpendicular manner because of this the stresses will follow the yield surface thus resulting in work hardening of the material. The unloading reloading parts of the stress-strain relationship exhibit the same behaviour as the elastic part of the model.

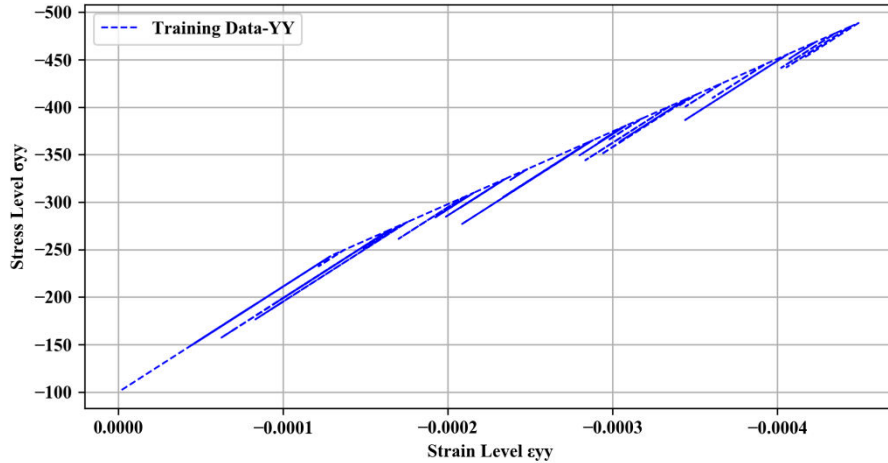


Figure 64 The work hardening model training dataset strain vs stress level

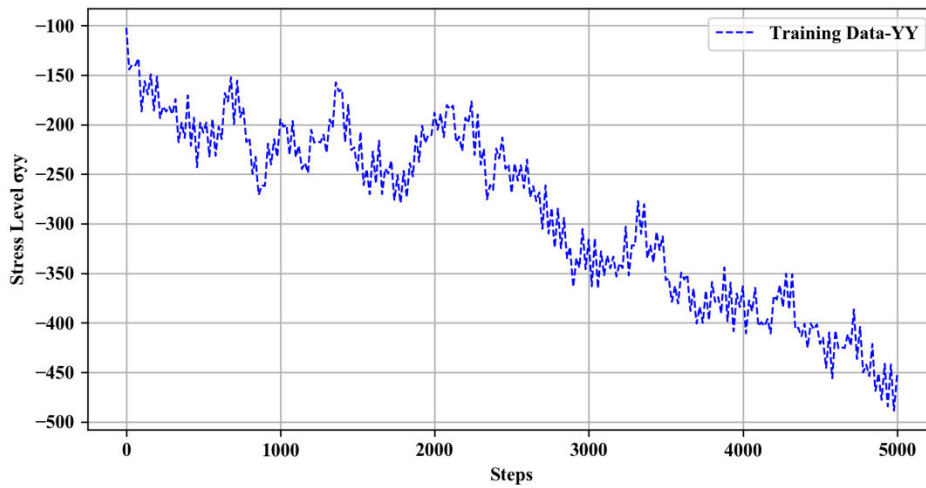


Figure 65 The work hardening model training dataset stress vs steps

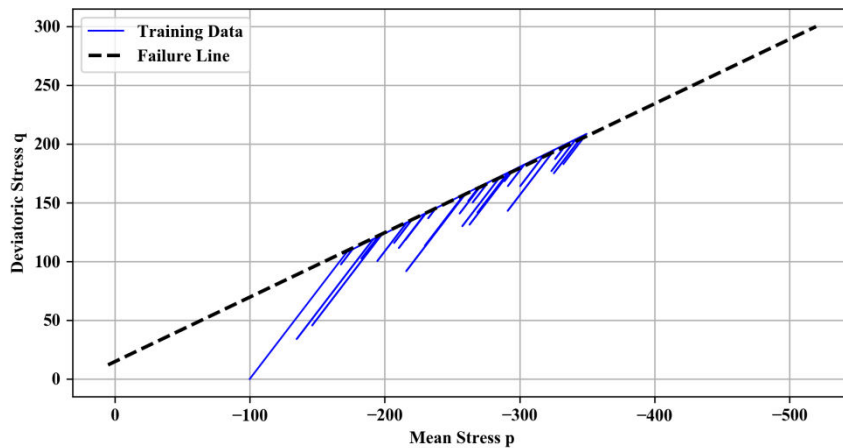


Figure 66 The work hardening model, training dataset p-q curve

### 4.2.3 Creation of Neural Networks

Creating material models as Neural Networks requires a combination of:

- Understanding the soil behaviour
- A clear problem definition
- An understanding of existing literature
- An understanding of how Neural Networks work
- Datasets that capture the behaviour that is modelled

Rather than creating one generic model an effort is made to create each of the Networks individually with Neural Network architectures that are relevant to each material soil model. In that way the inner workings of the Neural Network will be investigated in detail and in case the Networks fail to model these relationships the reason behind the failure as well as the way to remedy issues will be easier to identify.

#### 4.2.3.1 Linear-Elastic Model

The Linear Elastic model is relatively straight forward as it is described in chapter 4.2.1.1. This model is described by a linear relationship. To model the linear elastic behaviour by using a Neural Network a linear activation function will be chosen for the neurons. As it is observed in Figure 67 the model will start with a relatively simple architecture and then complexity will be added to the Network in terms of input and output directions. Starting with the Neural Network (a), that architecture describes the linear relationship between stresses and strains but only in the y direction. As more directions of outputs are added the weights of the network between the neuron and the output are responsible for modelling the linear relationship between each of the outputs and the input. Finally, in model (d) the full stress strain relationship can be observed. Two new neurons are added to represent the necessary connections from each input to each output. The choice of three neurons will be further explained and validated in chapter 4.3.2.

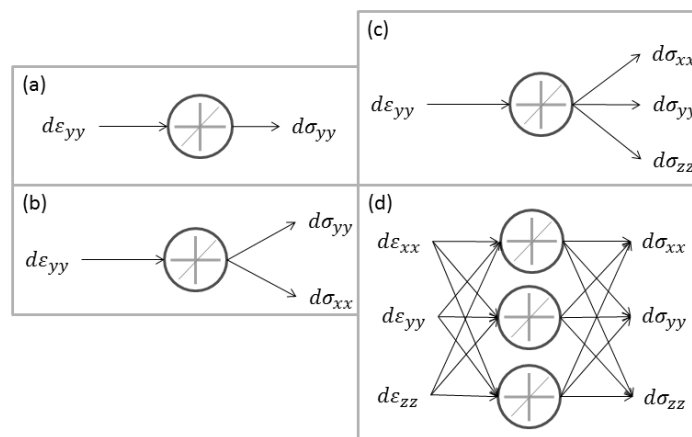


Figure 67 (a) Neural Network 1 direction output (b) Neural Network 2 directions as output (c) Three directions as output (d) Three directions as inputs and outputs of the Neural Network

#### 4.2.3.2 Linear-Elastic Perfectly-Plastic – Mohr Coulomb

As it was discussed in chapter 4.2.1.2 the Linear Elastic Perfectly Plastic model can be considered as two components in series with each other. In this case when the yield surface is reached the stress will remain in a constant level. To incorporate this behaviour in the Neural Network the inputs should include the stress level. Therefore, the inputs are the strain increment of the next step and the stress level of the previous step.

If the actual model is considered as component based, then the Neural Network can be thought as one as well. Thus, the Linear Elastic architecture presented in chapter 4.2.3.1 can be expanded by another layer of neurons and the additional stress level input discussed earlier. The activation function of the second layer should be a ReLU. The ReLU activation function has the ability of returning a linear function if inputs are larger than zero and a zero if the inputs are smaller or equal than zero. If we consider the scaling of the dataset the yield level are zeros (smallest number of the dataset). In that way the linear-elastic prediction will be made first and then depending on the stress level the prediction will be corrected if it reaches plasticity and return a single value for yield stress.

The main Neural Network architecture is presented in Figure 68. However, the structure of the ReLU layer has to be defined. The number of neurons as well as layers will be defined with by which carries the least error after training (Shahin M. , 2010). In this case the best architecture will be a combination of the smallest error with the smallest possible amount of neurons. Errors can be measured as mentioned in chapter 2.3.3. The next step is to validate the network by testing it against datasets with values not present in the training dataset. The inner workings of the network will be also tested to understand the networks behaviour during plasticity and elasticity. The outputs of each of the nodes are evaluated. To examine the network’s robustness extra testing datasets are created with different initial stresses. Those datasets represent different stress paths that still fall under the Linear-Elastic Perfectly-Plastic material model. The trained network will be tested against them and its performance will be evaluated. In addition, the testing datasets will be used to retrain the model. The effect of this is commented.

Finally, the Neural Network is tested against added noise. The noise data are created from a Gaussian distribution and then added to the already existing dataset. The Network is retrained with the intention of determining if the Neural Network recognises noise as a pattern in the dataset. Noise is added to the testing datasets to evaluate their performance with a Neural Network that is trained without the use of noise.

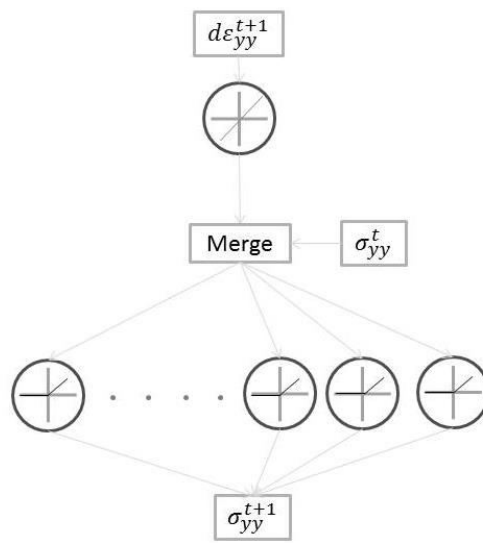


Figure 68 Main concept architecture Neural Network for the Mohr Coulomb model

#### 4.2.3.3 Linear-Elastic Work Hardening Model

In chapter 4.2.1.3 the work hardening stress-strain relationship is described. To produce a Neural Network equivalent to the linear elastic work-hardening model the continuum mechanics approach is considered. When the Neural Network and the material model put side by side (Figure 69 and Figure 70) it can be seen that the first elastic spring component is modelled neuron with a linear activation function. Moving to the friction component this relationship is modelled in a similar fashion as in chapter 4.2.3.2. Thus, consists of ReLU nodes and layers. Finally, a linear component which does not have any weight connections with the friction part is added. This part accounts for the linearity after the material reaches plasticity. The Neural Network is also in accordance with the work-hardening equation. As all parts of equation 22 are represented by the Neural Network.

$$\sigma = \begin{cases} E\varepsilon & \text{for } \sigma \leq \sigma_R \\ \sigma_R + E_t(\varepsilon - \sigma_R/E) & \text{for } \sigma > \sigma_R \end{cases} \quad (22)$$

Thus, the first step is to define a successful architecture for the Feedback prediction. The architecture with the more efficient number of nodes and layers will be selected. The next step is to validate trained Neural Network first in terms of the training dataset. Then the validation process can be resumed with computing the metrics of the testing dataset. At this staged it will be obvious if generalisation has occurred in for the Neural Network. The inner working of the Neural Network will be then discussed. That includes the weight contribution of each input as well as when each node become active during plasticity or elasticity.

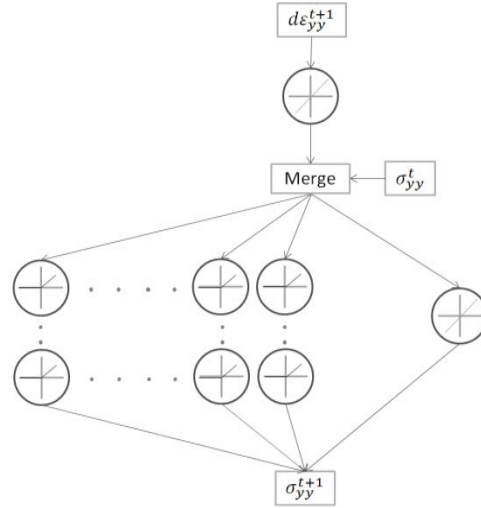


Figure 69 Main concept architecture of the Neural Network Work Hardening Model

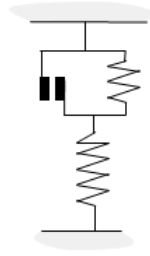


Figure 70 The components of the linear elastic work hardening model

### 4.3 Results Linear- Elastic Model

After training the Neural Networks the results can be evaluated for their accuracy. The generalisation ability of each network will be tested by creating testing datasets. Also the Neural Networks will be evaluated by performing a sensitivity analysis.

#### 4.3.1 Neural Network Model with One Direction as Input

The linear elastic model is essentially a linear relationship between the stresses and the strains. Thus, in this case the Neural Network will be considered as successful if the relationship between inputs and outputs has the same gradient as the Hooke's Law (chapter 4.2.1.1). In this example the Neural Network is trained by imposing increments of strain in one direction. Thus, strain increments  $d\epsilon_{xx}$  and  $d\epsilon_{zz}$  are zero and will not be included in the equations describing the stress strain relationship. This implies that the stresses will be described by equations 23, 24 and 25 . In the training dataset the Poisson's ratio is set to  $\nu=0.3$  and the Young's Modulus at  $E=1e+6 \text{ kN/m}^2$ . Thus the gradient of x and z direction will be  $576923.07 \text{ kN/m}^2$  and in the y direction  $1346153.84 \text{ kN/m}^2$ .

$$\sigma_{xx} = E \frac{1 - \nu}{(1 + \nu)(1 - 2\nu)} d\epsilon_{yy} \quad (23)$$

$$\sigma_{yy} = E \frac{\nu}{(1 + \nu)(1 - 2\nu)} d\epsilon_{yy} \quad (24)$$

$$\sigma_{zz} = E \frac{\nu}{(1 + \nu)(1 - 2\nu)} d\epsilon_{yy} \quad (25)$$

The input of the network needs to be scaled. In this case each feature will be scaled by its maximum absolute value. This process is called Normalization. The scaling will estimate and translate each feature individually

such that the maximal absolute value of each feature in the training set will be 1.0. The centre of the data is not shifted, and thus does not destroy the sparsity of the data.

#### 4.3.1.1 One Direction of Stress as Output

The first Neural Network in this case will be a model of equation 26. Thus, the first model only concerns the y direction of stress and strain. The input of the network is the strain increment and the output of the model is the stress increment. The fact that the output is a stress increment eliminates the possibility of errors occurring through a feedback prediction. This incremental prediction can be applied in this case since the linear elastic model will not reach failure, thus there is no need to have an indication of the stress level.

The network is observed in Figure 71. The Network only possesses two weights and biases are deactivated. The network is trained with the dataset presented in chapter 4.2.2.1. After training the network the final connection weights are obtained. In Table 7 the weights of the final trained network are shown. The relationship between stresses and strains from the Network can be calculated from equation 26. The scaler is defined as the absolute maximum value of the dataset (scaler =  $9.8e-4$ ). In this case only the inputs are scaled because the negative values and the proximity between the values of the does not help establish a connection between inputs and outputs.

Using equation 26 the success of the Network can be defined. The gradient between stresses and strains is  $1346153.84$ . From the Neural Network the gradient calculated from equation 26 is  $1350885.30$ . Thus the percentage error of the Neural Network is 0.3%.<sup>4</sup>

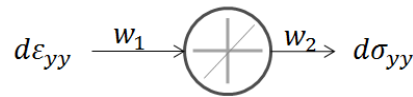


Figure 71 The linear-elastic Neural Network in one direction

Table 7 The weights of the trained Neural Network

Weights	
w1	36.41
w2	36.36

$$d\sigma_{yy} = w_1 w_2 \frac{d\epsilon_{yy}}{\text{scaler}} \quad (26)$$

The results of the training dataset can be observed in Figure 72 and in Figure 73. In those figures it is observed that the errors of the dataset are quite low. Thus the Neural Network was successful in uncovering the linear relationship between stresses and strains.

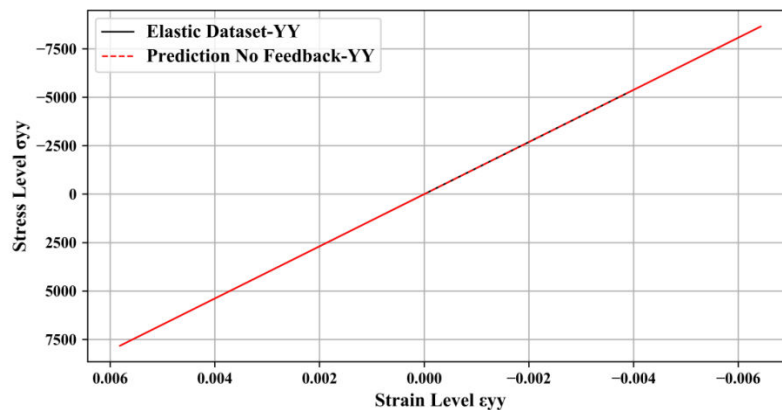


Figure 72 Diagram of linear stress and strain relationships

<sup>4</sup> In this case percentage error is defined as  $\text{percentage error} = \frac{\text{prediction} - \text{actual}}{\text{actual}} 100\%$

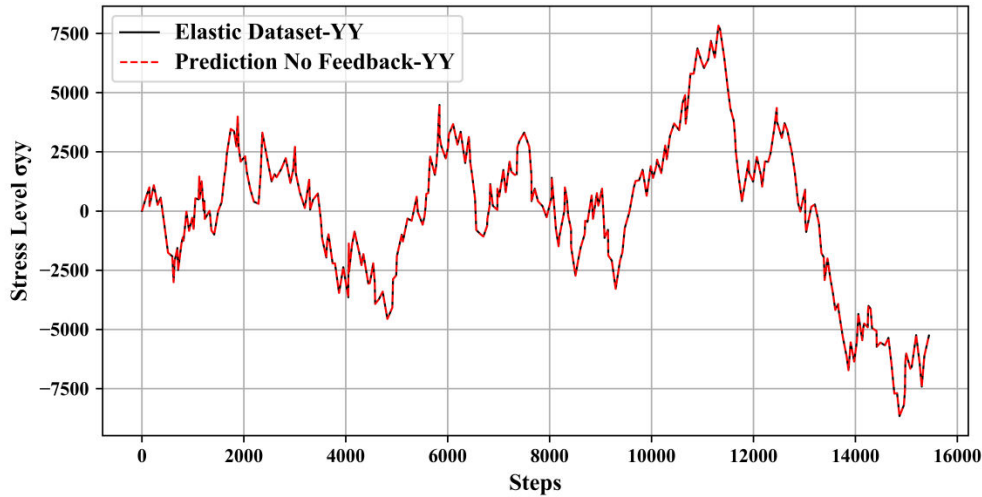


Figure 73 Stress versus steps for the training dataset

However, to complete the validation process different testing datasets are created. They are created in a similar random fashion as the training dataset (using the same material parameters) but they consist of less strain increments and different types of increment divisions through the steps. In PLAXIS the strain increment is divided to steps. Different step sizes are selected to test if the Neural Network has captured the behaviour of the soil model. The training can be characterized as successful since the Mean Absolute Errors of the testing are lower than the training dataset's Mean Absolute Error. This can be further validated by Figure 74, where all of the predictions present a linear 1 by 1 relationship with their respective outputs.

Table 8 Results of different testing dataset as Mean absolute error

	Steps in PLAXIS	Mean Absolute Error
Training Dataset	100	0.002122
Testing Dataset 1	100	0.001654
Testing Dataset 2	50	0.000540
Testing Dataset 3	20	0.000962
Testing Dataset 4	Random Uniform from 1 to 100	0.003617

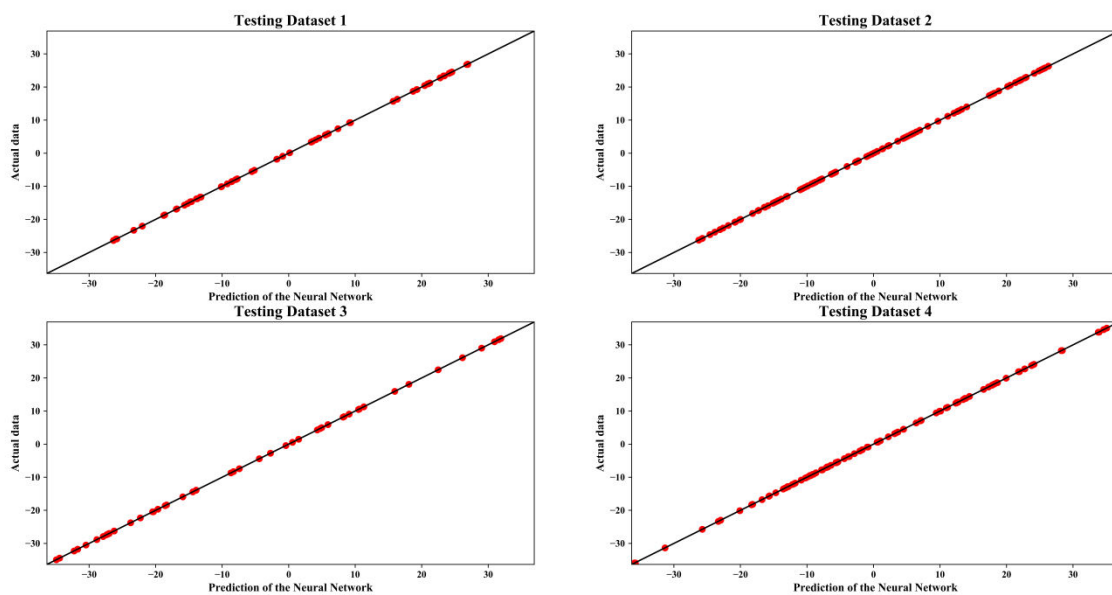


Figure 74 Different testing datasets in 1by 1 comparisons



### 4.3.1.2 Two directions as Outputs of the Neural Network

In this case the generalisation ability of the Linear Elastic Neural Network can be tested with attempting to stepwise construct the full stress-strain linear elastic matrix. Since modelling the linear elastic model in the y direction was successful the next step of this process is to add more directions gradually as an output to recreate the linear relationships described in equation 23 and 22. The network in Figure 75 will be used for the training. The scaling in this case will be the same as in chapter 4.3.1.1 thus scaler =  $9.8e-4$ . In equations and the mathematical relationships between stresses and strain as it is defined in the network can be seen. In table the results of the training can be observed. The training gave a percentage error of related to the gradient equal to 0.005% with for the y direction and for the x direction 0.003%. Overall these errors are acceptable if they validated through Figure 76 and Figure 77, where the prediction with the actual data coincide.

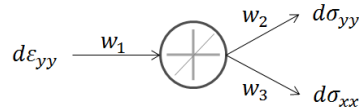


Figure 75 Neural Network with 2 directions as output

Table 9 Weights of the 2-direction Linear Elastic Model

w1	35.94
w2	36.84
w3	15.79

$$d\sigma_{yy} = w_1 w_2 \frac{d\epsilon_{yy}}{\text{scaler}} \quad (27)$$

$$d\sigma_{xx} = w_1 w_3 \frac{d\epsilon_{yy}}{\text{scaler}} \quad (28)$$

Table 10 Gradients of prediction and Hooke's Law

	Hooke's Law	Neural Network	Percentage Difference (%)
Gradient in the x	576917.66	576946.18	0.005
Gradient in the y	1346164.67	1346209	0.003

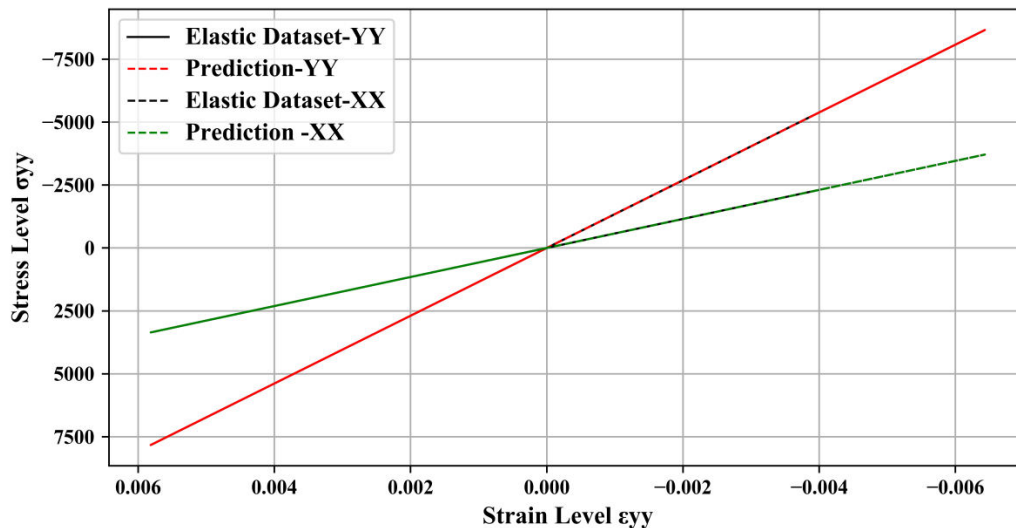


Figure 76 The training dataset linear relationship between stresses and strains

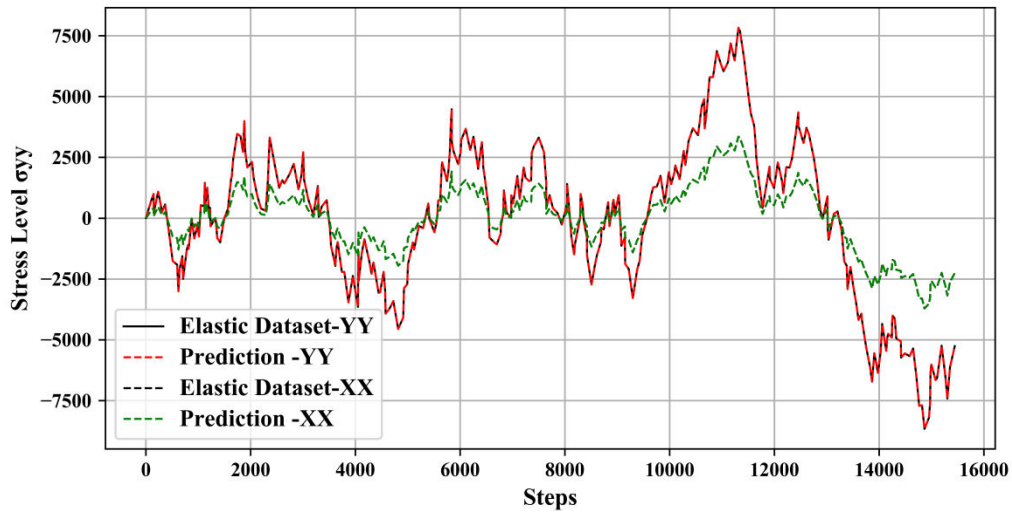


Figure 77 Training dataset step Vs stresses in both directions

However, the Neural Network needs to be further validated through the testing datasets. In Table 11 the mean absolute errors of the testing datasets are, almost for all the cases, smaller than the training ones. This can be further verified by Figure 78 and Figure 79 where the 1 by 1 linear relationship between the prediction and the actual data verify the success of the Neural Network.

Table 11 Results of different testing dataset as Mean absolute error

	Steps in PLAXIS	Mean Absolute Error in the y direction	Mean Absolute Error in the x direction
Training Dataset	100	0.002164	0.000848
Testing Dataset 1	100	0.001829	0.000705
Testing Dataset 2	50	0.000675	0.000225
Testing Dataset 3	20	0.001663	0.000357
Testing Dataset 4	Random Uniform from 1 to 100	0.003784	0.001544

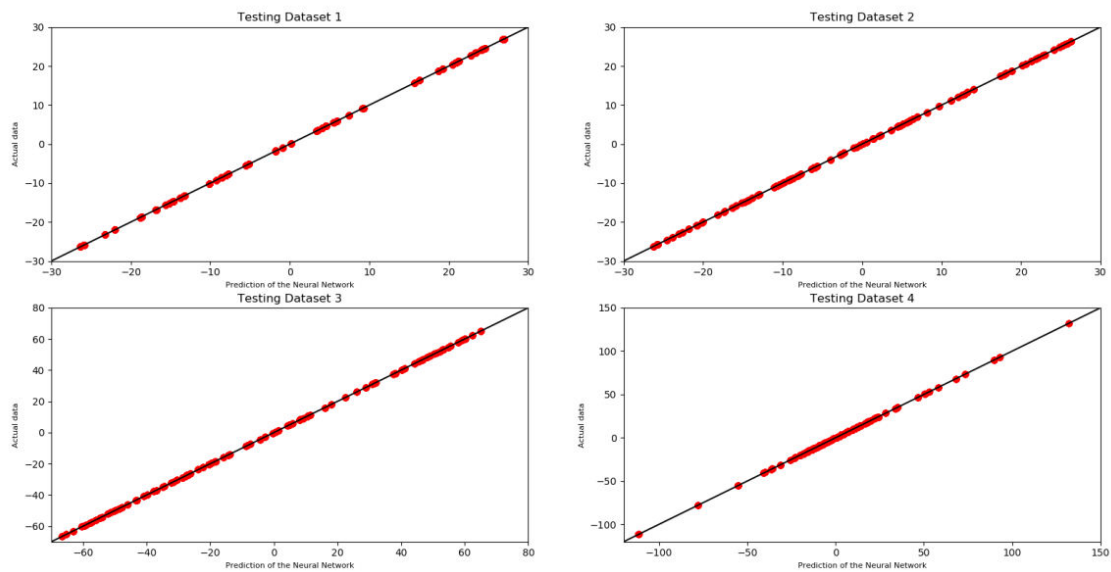


Figure 78 Testing Dataset results in the Y direction

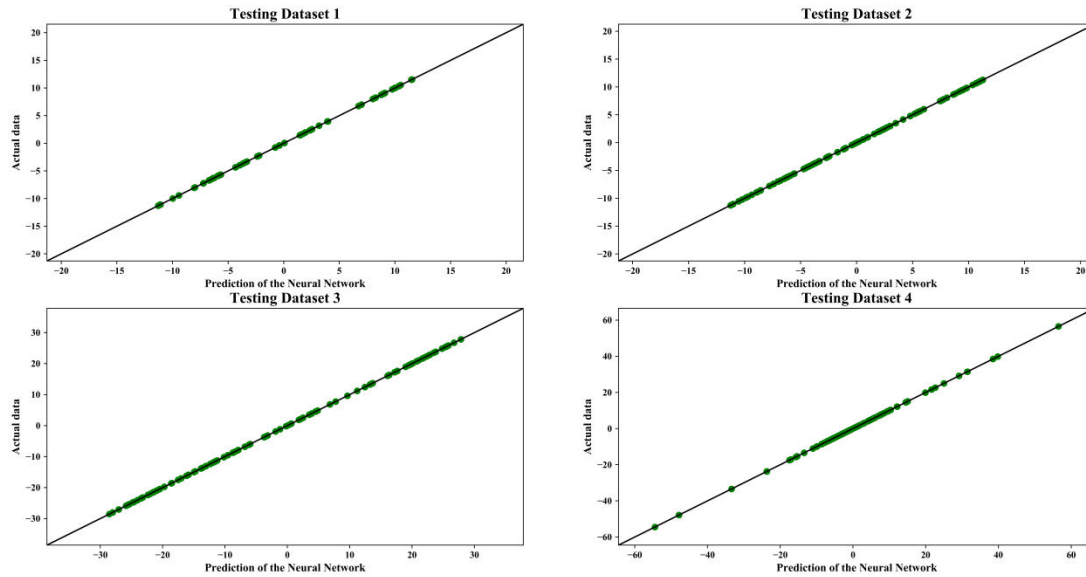


Figure 79 Testing Dataset results in the X direction

#### 4.3.1.3 Three directions as Outputs of the Neural Network

The next step in the linear elastic model is to try to add the third dimension as an output of the Neural Network model. The network in Figure 80 will be used for the training. The scaling in this case will be the same as in chapter 4.3.1.1 thus the scaler is:  $scaler = 9.8e-4$ . In equations 29, 30 and 31 the relationships between stresses and strains from the Neural Network can be derived. The gradient represented in these equations should be similar with the gradients of the equations 23, 24 and 25 which are derived from the Hooke's law. From Table 13 the gradients of each output can be observed as well as the errors between the Hooke's law and the Neural Network. From the table it is concluded that the Neural Network has been trained successfully. In Figure 81 and Figure 82 the fit of the Network for all the outputs can be observed.

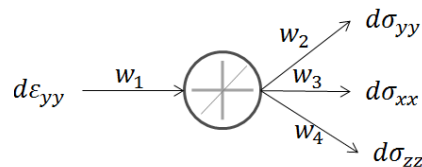


Figure 80 Neural Network with 3 directions as output

Table 12 Weights of the 2-direction Linear Elastic Model

w1	34.28
w2	38.63
w3	16.55
w4	16.55

$$d\sigma_{yy} = w_1 w_2 \frac{d\varepsilon_{yy}}{scaler} \quad (29)$$

$$d\sigma_{xx} = w_1 w_3 \frac{d\varepsilon_{yy}}{scaler} \quad (30)$$

$$d\sigma_{zz} = w_1 w_4 \frac{d\varepsilon_{yy}}{scaler} \quad (31)$$

Table 13 Gradients of prediction and Hooke's Law

	Hooke's Law	Neural Network	Percentage Difference (%)
Gradient in the x	1346195	1346165	0.002
Gradient in the y	576949.5	576917.7	0.005
Gradient in the z	576945.7	576917.7	0.005

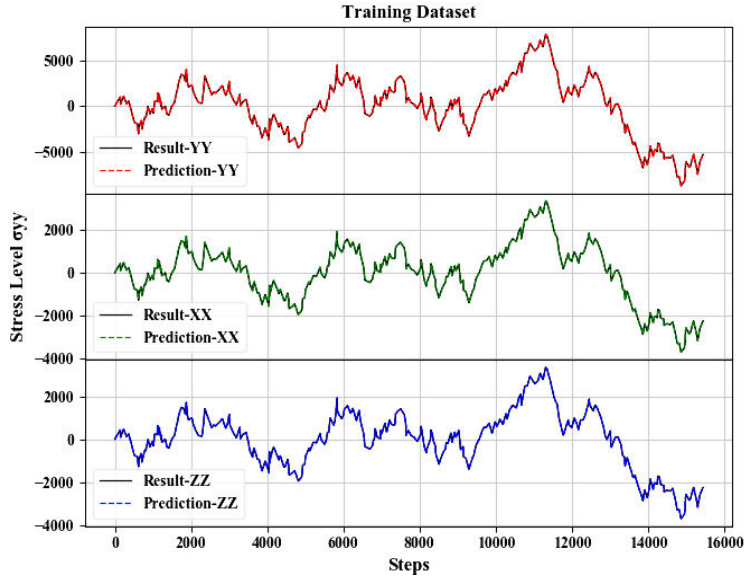


Figure 81 The results of the Neural Network with 3 outputs. Stress Vs Steps



Figure 82 The results of the Neural Network with 3 directions as outputs. Stress Vs Strain

The generalisation ability of the Neural Network is finally validated through the testing datasets as it was done in chapters 4.3.1.1 and 4.3.1.2. In Table 14 the fact that the Neural Network has uncovered the general stress-strain relationship is backed up by the low errors of the testing datasets, this can be further validated from Appendix A. In Figure 127, Figure 128, Figure 129 the 1 by 1 linear relationship between the prediction and the actual data verify the accuracy of the Neural Network.

Table 14 Results of different testing dataset as Mean absolute error

	Steps in PLAXIS	Mean Absolute Error in the y direction	Mean Absolute Error in the x direction	Mean Absolute Error in the z direction
Training Dataset	100	0.002149	0.00098	0.000954
Testing Dataset 1	100	0.001772	0.000827	0.000796
Testing Dataset 2	50	0.000656	0.000334	0.000309
Testing Dataset 3	20	0.001541	0.000881	0.000786
Testing Dataset 4	Random Uniform from 1 to 100	0.003743	0.001665	0.001638

#### 4.3.2 Neural Network Model with Three Directions as Input

Since the three previous models were successful in modelling the linear elastic behaviour the model can be theoretically expand to full stress strain relationship in the three directions (xx, yy, zz). In this case the strains in each step are imposed in these three directions and they are selected randomly from a uniform distribution. The stresses and strains have linear relationships governing them. From equation 32 the equations 33, 34 and 35 are derived. These equations represent the modelling target of the Neural Network. If the Neural Network successfully models them then generalisation of the model has been achieved.

$$\begin{bmatrix} d\sigma_{xx} \\ d\sigma_{yy} \\ d\sigma_{zz} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu \\ \nu & 1-\nu & \nu \\ \nu & \nu & 1-\nu \end{bmatrix} \begin{bmatrix} d\varepsilon_{xx} \\ d\varepsilon_{yy} \\ d\varepsilon_{zz} \end{bmatrix} \quad (32)$$

$$d\sigma_{xx} = \frac{E}{(1+\nu)(1-2\nu)} (d\varepsilon_{xx}(1-\nu) + (d\varepsilon_{yy} + d\varepsilon_{zz})\nu) \quad (33)$$

$$d\sigma_{yy} = \frac{E}{(1+\nu)(1-2\nu)} (d\varepsilon_{yy}(1-\nu) + (d\varepsilon_{xx} + d\varepsilon_{zz})\nu) \quad (34)$$

$$d\sigma_{zz} = \frac{E}{(1+\nu)(1-2\nu)} (d\varepsilon_{zz}(1-\nu) + (d\varepsilon_{xx} + d\varepsilon_{yy})\nu) \quad (35)$$

##### 4.3.2.1 Neural Network with Three Directions as output

So far the hidden layer was been composed of only one neuron. In this case since the inputs are three it can be assumed that the one neuron will not be able to capture the linear connections between inputs and outputs. To determine the optimum number of neurons in the hidden layer training with 1, 2, 3 and 4 nodes is initiated. The errors of the Neural Networks are documented and the final Loss per Epoch function can be graphed. From Figure 83 the optimum number of nodes in the hidden layer can be defined. In this case both the 3 and 4 noded Neural Networks end up with the same final error. However, the 3 noded hidden layer will be selected as it leads to the same result by using less neurons. Thus, the final form of the network can be defined in Figure 84.

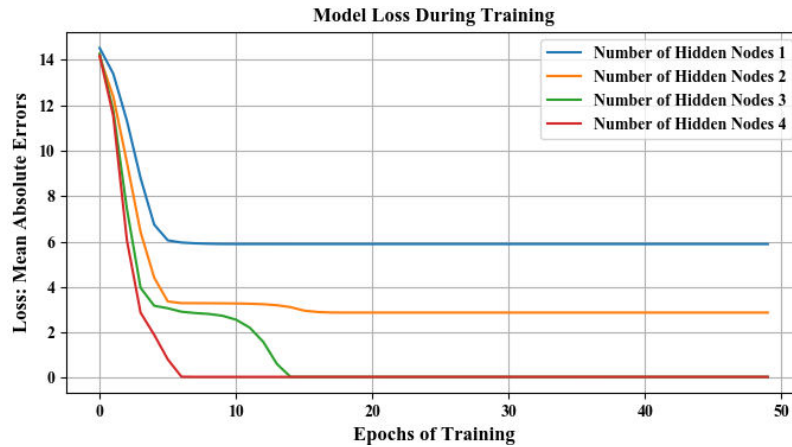


Figure 83 Training results for different number of nodes

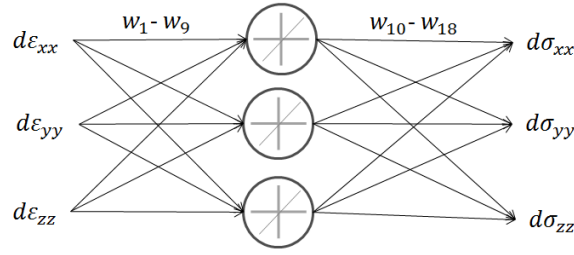


Figure 84 Neural Network Linear Elastic 3 inputs 3 outputs

In this Neural Network the biases are deactivated. Between the input and the hidden layer 9 weights exist ( $w_1$  to  $w_9$ ), between the hidden layer and the output another 9 weights exist ( $w_{10}$  to  $w_{18}$ ). From the feed-forward network definition the mathematical relationship between inputs and outputs can be defined. In equations the relationship between stresses and strains can be observed. The aim of the training of the Neural Network is to model the behaviour between stresses and strains as it can be observed in the linear-elastic Hooke's Law. Thus, the weight matrix of the Neural Network should be the same as the Linear Elastic one. After the training of the Neural Network the weight matrix is produced. The material matrix can be also calculated with  $\nu=0.3$  and  $E=1e+06 \text{ kN/m}^2$ . In Table 15 the resulting matrices can be observed from the results of the Neural Network model it is concluded that the relationship between stresses and strains was modelled successfully. Finally, the actual fit of the results can be seen in Figure 85 as further validation.

$$\begin{bmatrix} d\sigma_{xx} \\ d\sigma_{yy} \\ d\sigma_{zz} \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix} \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} d\varepsilon_{xx} \\ d\varepsilon_{yy} \\ d\varepsilon_{zz} \end{bmatrix} \quad (36)$$

$$\begin{bmatrix} d\sigma_{xx} \\ d\sigma_{yy} \\ d\sigma_{zz} \end{bmatrix} = \begin{bmatrix} w_{10}w_1 + w_{13}w_2 + w_{16}w_3 & w_{10}w_4 + w_{13}w_5 + w_{16}w_6 & w_{10}w_7 + w_{13}w_8 + w_{16}w_9 \\ w_{11}w_1 + w_{14}w_2 + w_{17}w_3 & w_{11}w_4 + w_{14}w_5 + w_{17}w_6 & w_{11}w_7 + w_{14}w_8 + w_{17}w_9 \\ w_{12}w_1 + w_{15}w_2 + w_{18}w_3 & w_{12}w_4 + w_{15}w_5 + w_{18}w_6 & w_{12}w_7 + w_{15}w_8 + w_{18}w_9 \end{bmatrix} \begin{bmatrix} d\varepsilon_{xx} \\ d\varepsilon_{yy} \\ d\varepsilon_{zz} \end{bmatrix} \quad (37)$$

$$\begin{bmatrix} d\sigma_{xx} \\ d\sigma_{yy} \\ d\sigma_{zz} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu \\ \nu & 1-\nu & \nu \\ \nu & \nu & 1-\nu \end{bmatrix} \begin{bmatrix} d\varepsilon_{xx} \\ d\varepsilon_{yy} \\ d\varepsilon_{zz} \end{bmatrix} \quad (38)$$

Table 15 Weight and Linear Elastic Matrix comparison

Linear Elastic Matrix			Neural Network Weight Matrix			Percentage Error between Matrices (%)		
1346154	576923.1	576923.1	1345253	576538.4	576467.1	0.066	0.066	0.079
576923.1	1346154	576923.1	576383.2	1345218	576472	0.093	0.069	0.078
576923.1	576923.1	1346154	576451.3	576531.6	1345149	0.081	0.067	0.074

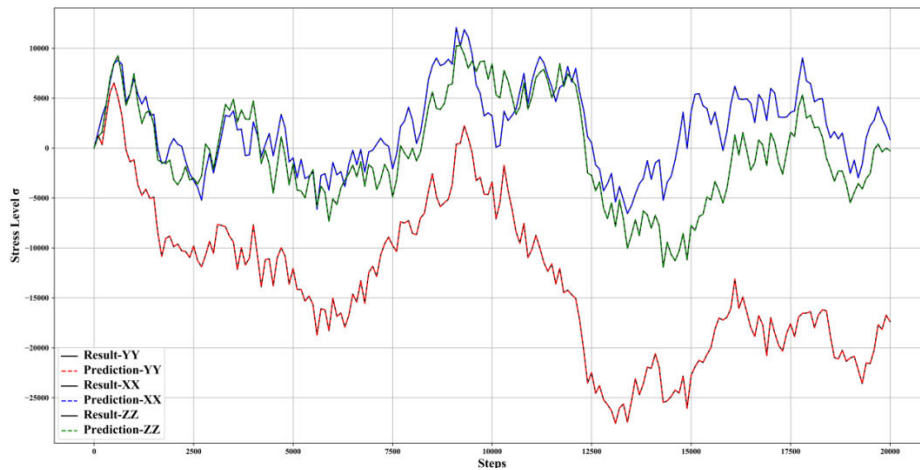


Figure 85 Results of stress level Vs Steps in three directions

Finally, to fully validate the results of the model the Neural Network's performance is calculated with 4 different testing datasets. In Table 16 the mean absolute error for all the datasets is shown. The testing dataset have smaller errors than the training which gives the impression that the generalisation of the neural network model is achieved. This is further validated by Figure 86 to Figure 88 where the 1 by 1 relationship of prediction and actual data are shown. In all the datasets and directions the data are close to the linear line and thus there are negligible errors between prediction and actual results.

Table 16 Results of different testing dataset as Mean absolute error

	Steps in PLAXIS	Mean Absolute Error in the y direction	Mean Absolute Error in the x direction	Mean Absolute Error in the z direction
Training Dataset	100	0.035796	0.072889	0.03601
Testing Dataset 1	100	0.002772	0.002947	0.002314
Testing Dataset 2	50	0.001999	0.004165	0.003149
Testing Dataset 3	20	0.005361	0.009532	0.00831
Testing Dataset 4	Random Uniform from 1 to 100	0.002033	0.004171	0.00343

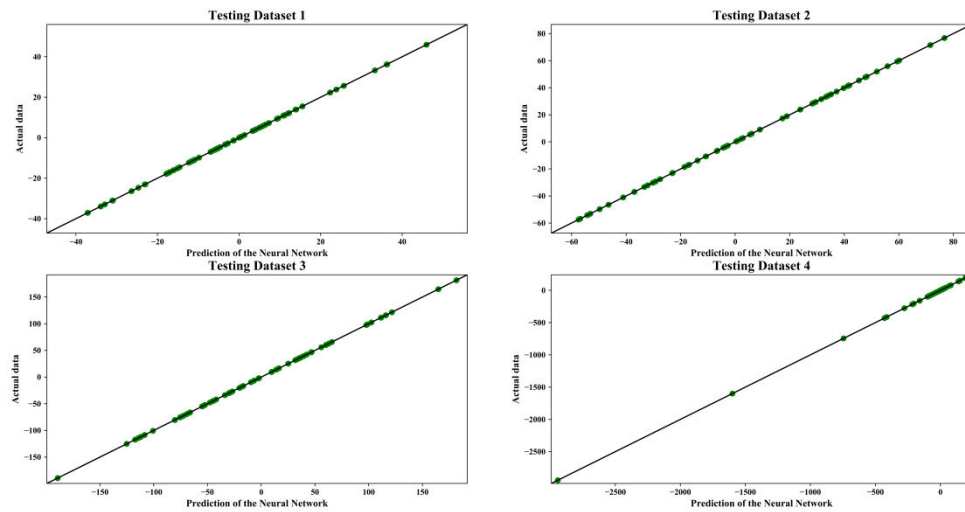


Figure 86 The results of testing datasets in the direction XX

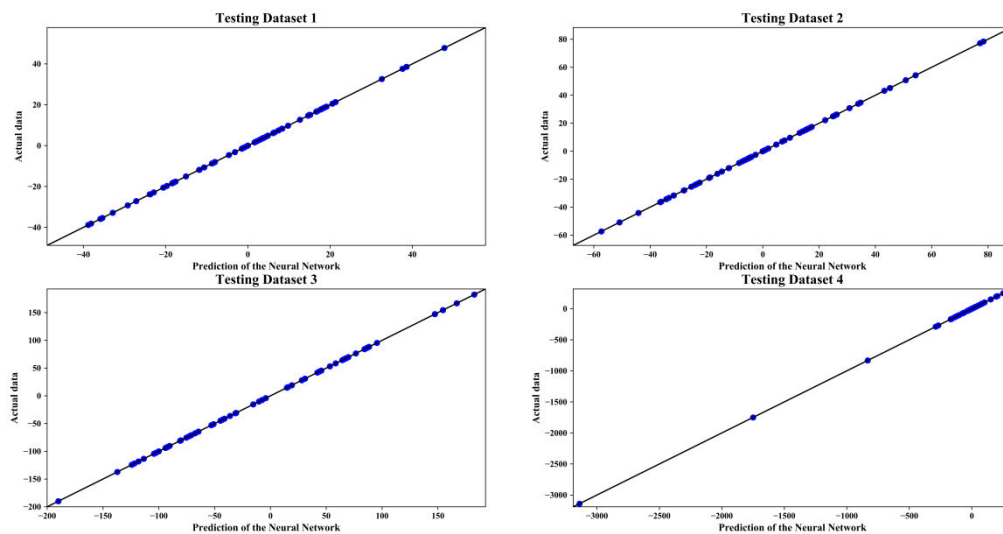


Figure 87 The results of testing datasets in the direction YY

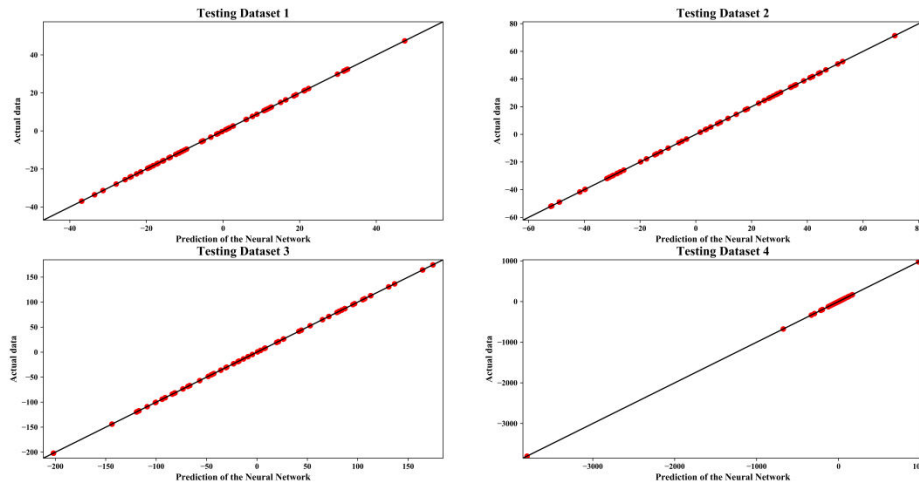


Figure 88 The results of testing datasets in the direction ZZ

## 4.4 Linear- Elastic Perfectly Plastic Model

### 4.4.1 Defining Network Architecture and Training Results

The Linear-Elastic Perfectly-Plastic model will be modelled as a Neural Network as it was discussed in chapter 4.2.3.2. Therefore, the first step is the determination of the architecture of the ReLU layer of the Neural Network. The Neural Network is trained by using different architectures in the ReLU layer. The results can be observed in Figure 89. From the figure it is observed that the smallest errors belong to the two layered 2 noded architecture or the three layers with 2 nodes architecture. The two layers with 2 nodes will be chosen as they represent shallowest network.

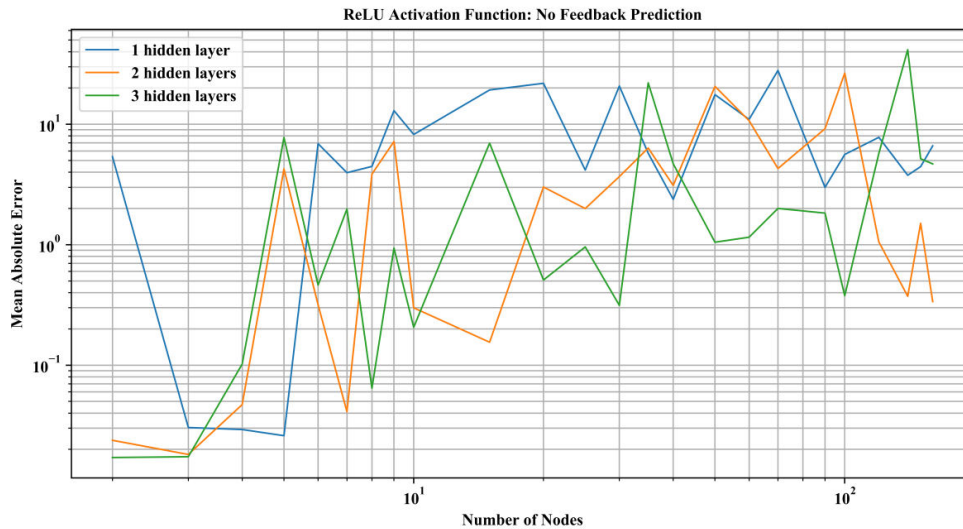


Figure 89 The errors of the ReLU layer

In Figure 90 the final Neural Network is shown. In this figure the connections are sized according to the weights. From the figure it can be seen that the stress level has a greater contribution than the strain increment in the network. To quantify how the inputs are connected to the outputs the Relative Contribution of each input can be calculated. Table 17 verifies what was seen in the figure. Thus, the stress level has a greater contribution to the output than the strain increment.

Table 17 The Relative Contribution of each input

RC1	RC2
strain input	stress input
4.09%	95.9%



To further validate and visualise the relative contribution to the Neural Network a sensitivity analysis will be conducted, this will be accomplished in a similar fashion as in chapters 3.3.2.2 and 3.3.3.1. This is computed by varying one of the inputs while the other one remains at a constant percentile value. In Figure 91 the results are observed. From the first plot the effect of the stress level input can be observed. Here the stress level follows a linear relationship until it reaches a plateau when stress inputs are smaller than  $-143 \text{ kN/m}^2$ . This implies that the Neural Network was successful in uncovering the ultimate stress level. However, it can be also concluded that the percentile of the strain increment input has little effect on the final result. When reviewing the second plot of the figure it can be observed that the percentile of stress input has a large influence on the output of the Neural Network. When only one percentile is displayed for both cases the relationship when the strain is varied can be observed closely (Figure 91). The strain input has a positive relationship with the stress output. This means that a positive strain increment will result in an increase of the stress level output. In that way the Network was successful in identifying the loading unloading behaviour.

In addition, the actual results can be observed in Figure 93 and Figure 94. In these figure the fit of the feedback prediction (chapter 3.2.3) and regular prediction can be seen. Both of them perform adequately. In this case for modelling the stress-strain relationship only the feedback prediction is of interest. To have a better understanding of the accuracy of the model the maximum and minimum absolute difference will be calculated as the dataset's errors are not quite visible in Figure 93 and Figure 94. The minimum absolute difference is  $1.99\text{e-}6$  and the maximum absolute difference is  $0.09$ . In the order of magnitude of this specific problem this error is acceptable.

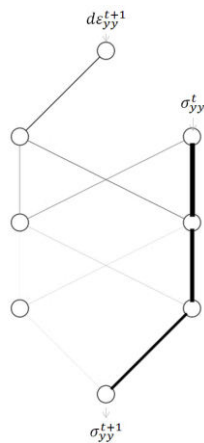


Figure 90 The final training of the Neural Network larger weights are sized accordingly

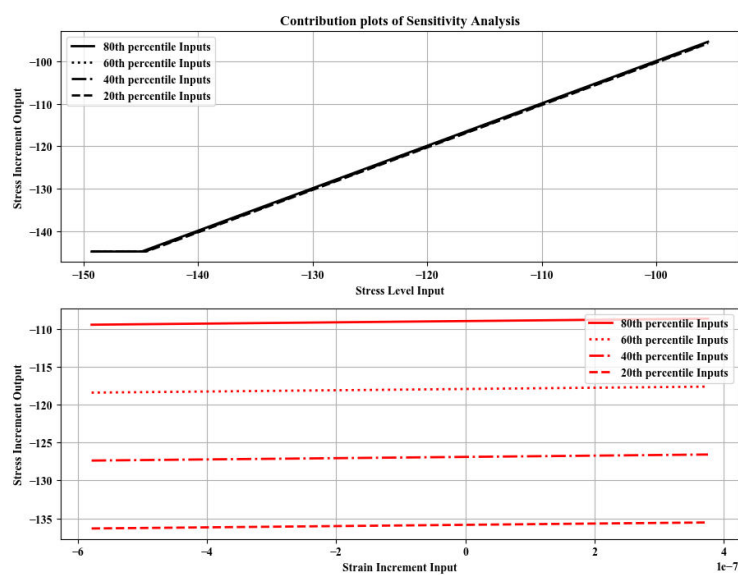


Figure 91 Contribution Plots from the sensitivity analysis illustrating the neural networks response curves to changes in each input with all the variables held constant in different percentiles.

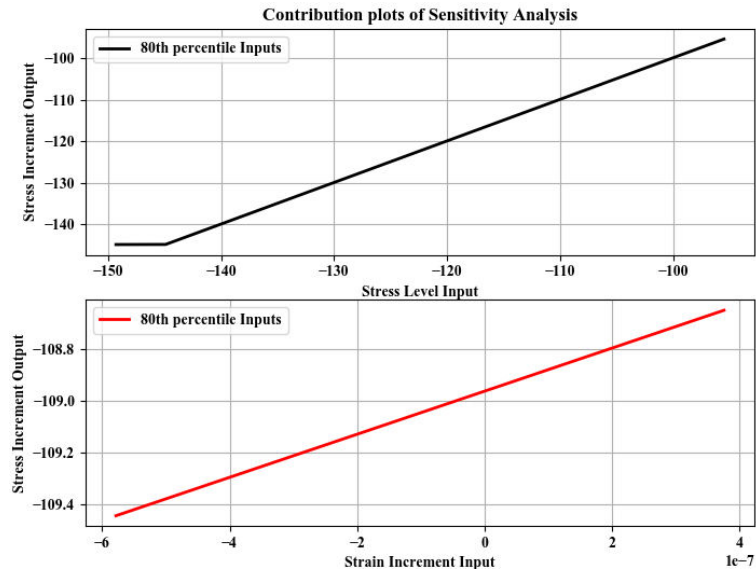


Figure 92 Contribution Plots from the sensitivity analysis illustrating the neural networks response curves to changes in each input with all the variables held constant in the 80th percentile.

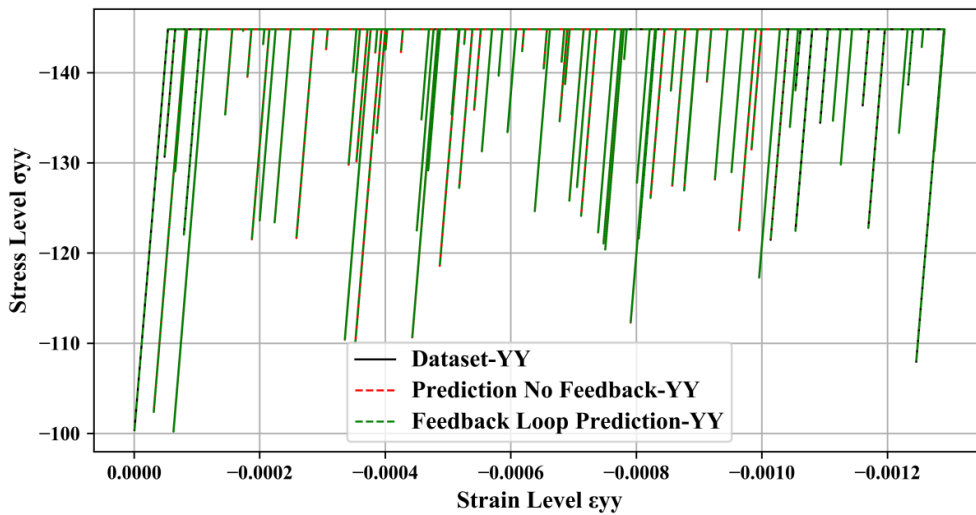


Figure 93 The Training Datasets Results with and without feedback prediction.

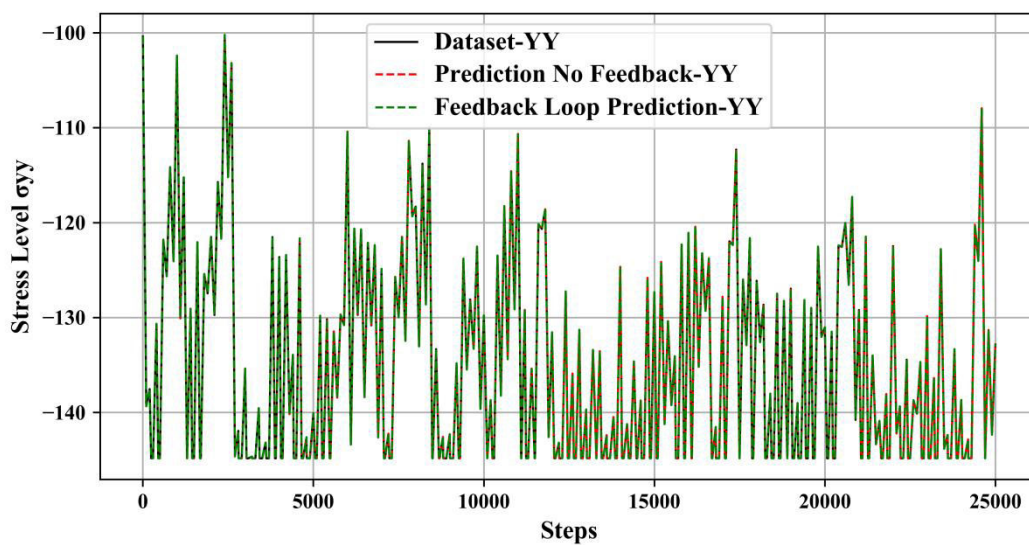


Figure 94 The Training Datasets Results with and without feedback prediction.

However, to fully understand how the Neural Network of linear-elastic perfectly-plastic model performs a link must be made between the inputs of the network the result and the output of each node. As it is described in chapter 2.1.3 the output of each node is the sum of weighted inputs plus a bias passed through an activation function. To understand the behaviour of the network the output of each node will be evaluated. The first step is to identify the output of the network reacts to different inputs in order to link the perfectly plastic behaviour with network inputs. It can be seen that the output of the network follows a ReLU trend (Figure 95). Thus, for stress inputs smaller than  $-143kN/m^2$  the output of the Network will always be the  $-143kN/m^2$ . That means that the network has uncovered the ultimate yield stress level.

The final step is to identify which neurons fire when the input is plastic and which when the input is elastic. For the purpose of these observations the strain is kept at a constant value. The output of each node is calculated. In Figure 96 the possible outputs of the nodes while changing the stress input are shown. In the figure it can be seen that plasticity occurs when node N3 and node N5 have an output of zero. That is the reason behind the results of Figure 95.

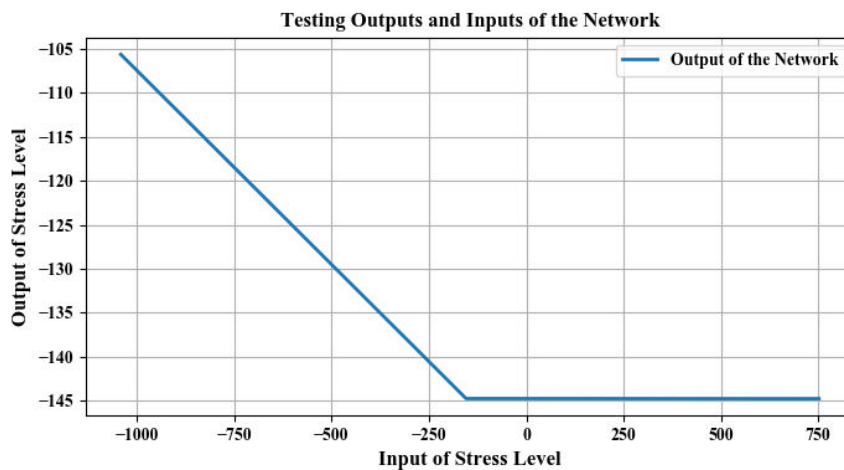


Figure 95 Testing the results of the Network

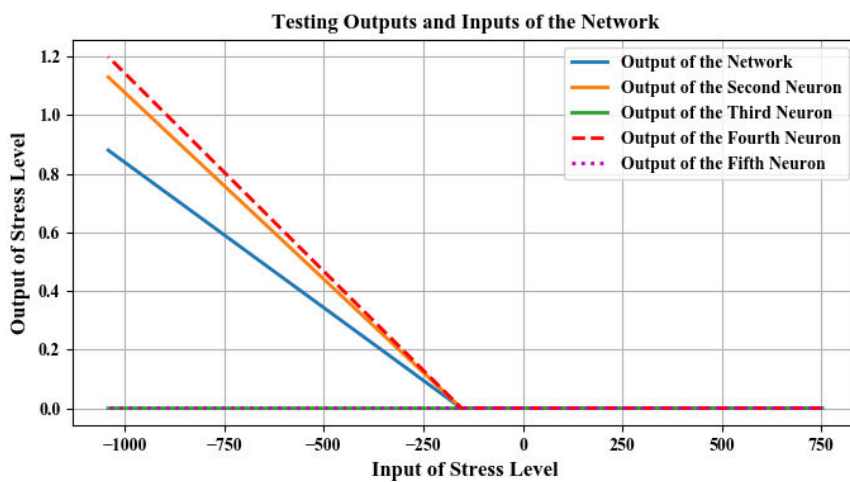


Figure 96 Outputs of each nodes

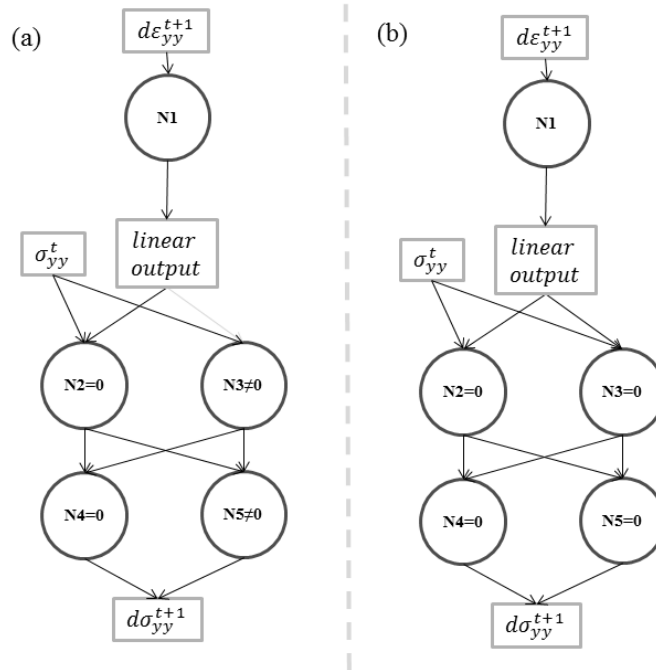


Figure 97 (a) Nodes outputs when elasticity is occurring (b) Outputs of each node when plasticity is occurring

#### 4.4.2 Validation of Neural Network

The validation process in this case will constrict of, firstly, validating against datasets with the same starting stress level. Then the validation process will include datasets with different initial stresses and finally noise will be incorporated in both testing and training of the dataset.

##### 4.4.2.1 Validation with Similar Datasets

The next step in the validation process is to use testing datasets to check if the Neural Network achieved generalisation. The testing datasets are created at PLAXIS in a similar fashion as in chapters 4.3.1.1, 4.3.1.2 and 4.3.1.3. In Table 18 the results of the testing prediction are appended. The testing results range within acceptable errors so the network has uncovered the relationship between stresses and strains. The results can be observed in detail in Figure 98 and Figure 99.

Table 18 Results of different testing dataset as Mean absolute error

	Steps in PLAXIS	Mean Absolute Error in the y direction	Minimum Absolute difference	Maximum Absolute Difference
Training Dataset	100	0.017425	1.99e-06	0.09
Testing Dataset 1	100	0.044726	5.13E-06	0.23
Testing Dataset 2	50	0.006943	2.36e-07	0.07
Testing Dataset 3	20	0.014159	5.13e-06	0.13
Testing Dataset 4	Random Uniform from 1 to 100	0.031198	5.13e-06	0.12

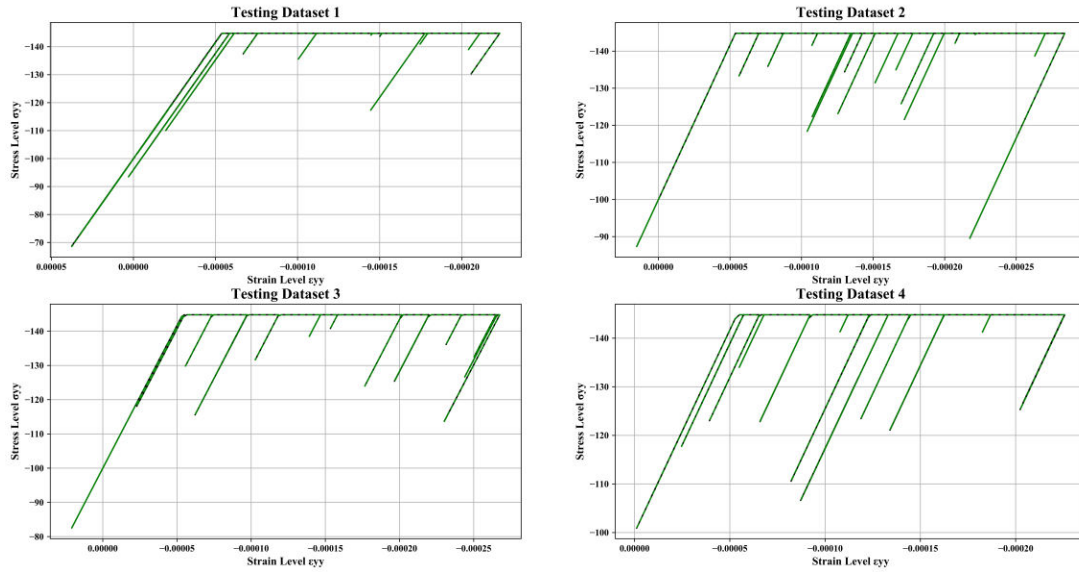


Figure 98 Testing Datasets results Strain Level Vs Stress Level

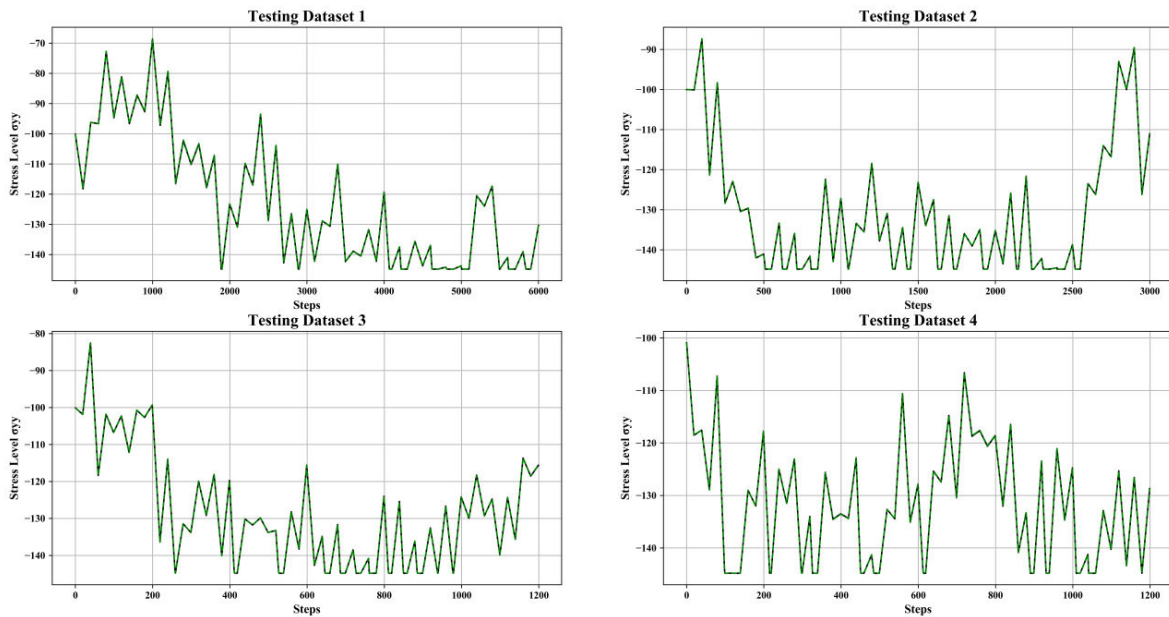


Figure 99 Testing Datasets results Stress Level Vs Steps

#### 4.4.2.2 Testing the Generalisation Ability – Different Initial Stresses

The trained Neural Network for Perfect Plasticity needs to be further evaluated for its robustness. The first step is to evaluate the model when the initial stresses are different from the training dataset. In the training dataset the starting stress vector is  $[\sigma_{xx} \ \sigma_{yy} \ \sigma_{zz}] = [-100 \ -100 \ -100] \text{ kN/m}^2$ . The initial stress vectors can be observed in Table 19. What is expected in this case is that the ultimate stress reached will be different in each case. It can be observed from Figure 100 that the p-q relationship will result in a different ultimate stress level.

Table 19 Testing Datasets with new initial Stresses

	$\sigma_{xx}$ (kN/m <sup>2</sup> )	$\sigma_{yy}$ (kN/m <sup>2</sup> )	$\sigma_{zz}$ (kN/m <sup>2</sup> )
Test Dataset 1	0	0	0
Test Dataset 2	-60	-60	-60
Test Dataset 3	-80	-80	-80
Test Dataset 4	-200	-200	-200

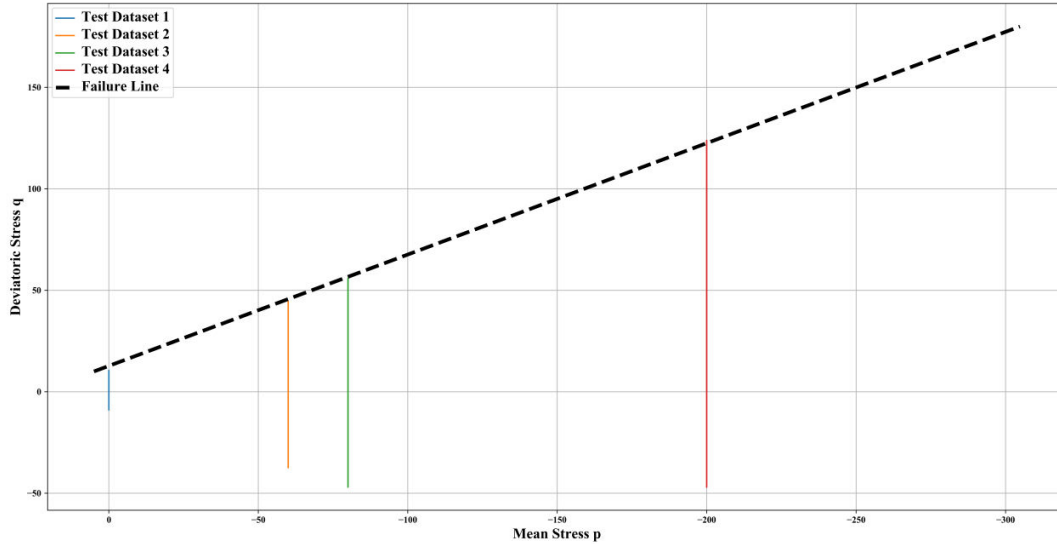


Figure 100 p q space of the test datasets

Each dataset is standardised by using the same maximum and minimum values of the training dataset (page 45). The bounds of the testing dataset after scaling can be observed in the Table 20. The table suggests that scaling with the minimum and maximum values of the training dataset results in data that are out of the [0,1] bounds. Since the network is trained for values between 0 and 1 the result is expected to have low accuracy. Specifically in the Testing Dataset 1 the stresses range between 3 and 3.35 which suggest that an ultimate stress level will not be reached in the ReLU layer. The same can be suggested for testing datasets 2 and 3. Finally, in the testing dataset 5 the stress level inputs and outputs are negative which might lead to the misinterpretation of the network assuming that the soil element of the dataset is in plasticity.

Table 20 The bounds of the scaled Testing Datasets

	Input Stain increment	Input Stress Level	Output Stress Level
Testing Dataset 1	0 to 1	3 to 3.35	3 to 3.35
Testing Dataset 2	0 to 1	1.2 to 2.4	1.2 to 2.4
Testing Dataset 3	0 to 1	0.6 to 2.2	0.6 to 2.2
Testing Dataset 4	0 to 1	-3 to -0.5	-3 to -0.5

Observing the results the issues of the Neural Network can be uncovered (Figure 101, Figure 102). The Neural Network fails to predict the correct ultimate stress level when the soil is in compression. The same value is the results in all the graphs (Figure 101). That value is  $-143 \text{ kN/m}^2$ , this value is the ultimate stress level of the training dataset). This confirms that the Neural Network has this ultimate value as the maximum output. In addition, the Neural Network fails to predict the ultimate stress during tension. This is expected as this behaviour was not captured during training.

Specifically, in Figure 101 Datasets 1 and 2 fail to predict the failure surface of the tension, that leads to a miscalculation of the stresses. Datasets 3 and 4 fail because the yield surface is not uncovered in the correct stress level during compression. However, the scaling and the fact that the yield surface is not recognised seem to be the biggest issues in this case. To resolve these issues first the standardisation of each training dataset can be achieved by scaling the dataset with the maximum and minimum contained in them. Secondly, the network (Figure 68) will be retrained with these values. In that way the Network's architecture can be tested for different initial stresses.

The Neural Network (Figure 68) is retrained by using the data of each testing dataset. That will result in 4 different networks with different weights and biases. Each dataset is scaled individually with values between 0 and 1. The results can be observed in Figure 103 and Figure 104. Generally, the upper limit of the yield surface is correctly discovered by the Neural Network after training. However, the ultimate stress level in terms of tension is not identified in the same way. However, the Neural Networks have modelled the limit with a type of polynomial function. That means that the yield surface is identified but a new architecture needs to be created to take into account the tension side of the yield surface. Finally, the general unloading-reloading behaviour is captured adequately in the graphs. This concludes that the Network was able to uncover the general loading-unloading-reloading behaviour of the dataset.

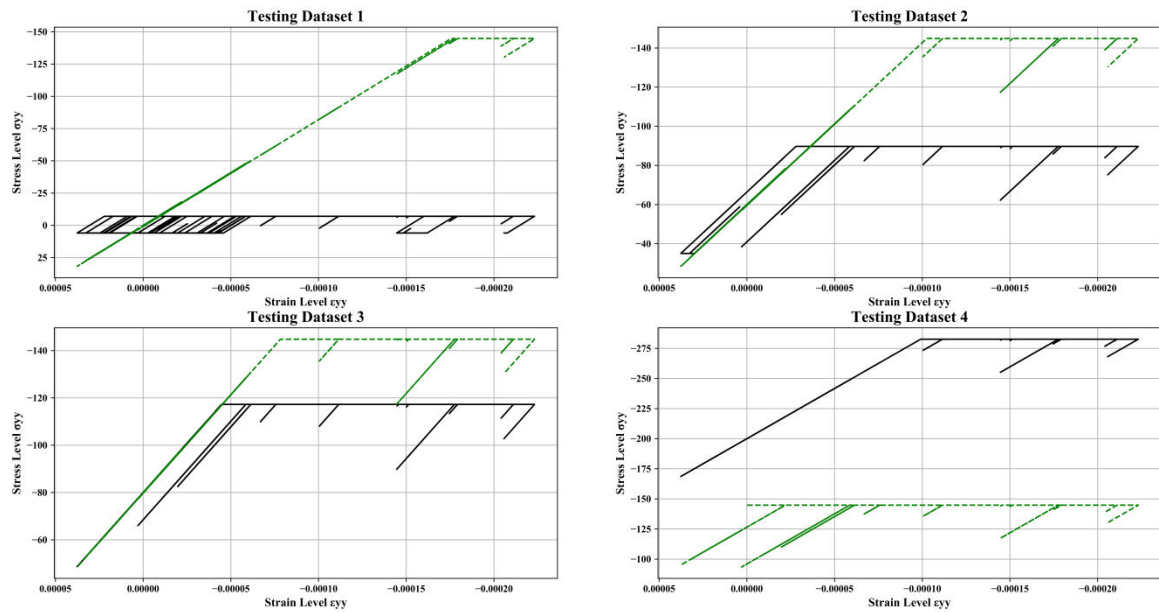


Figure 101 Results of the testing datasets stress Vs strain Level

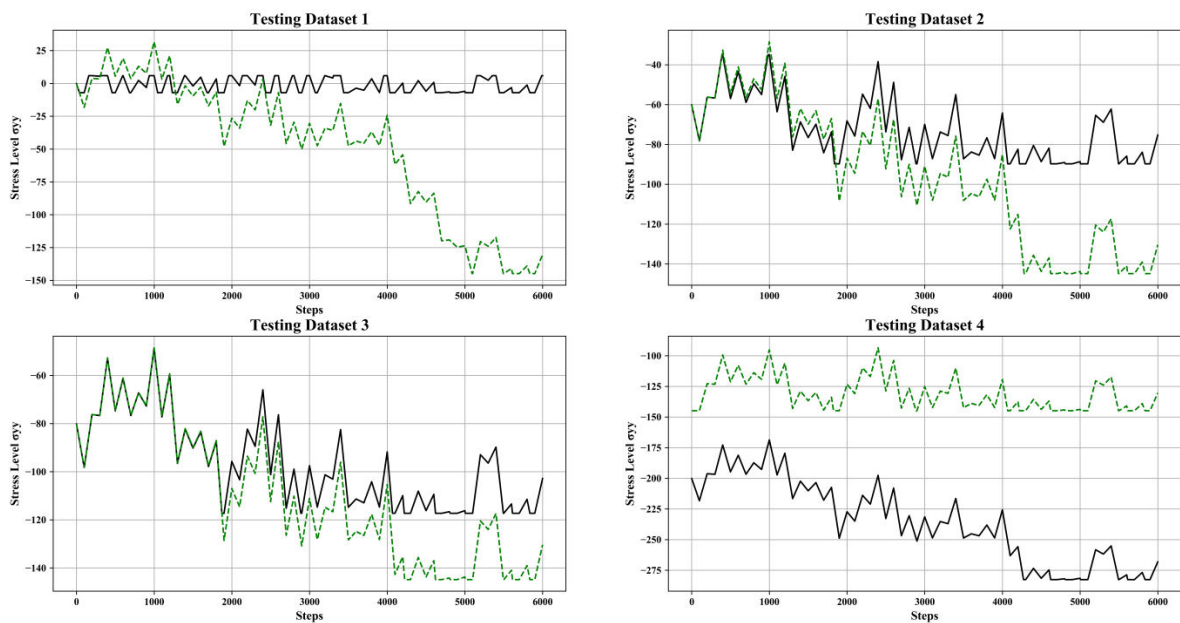


Figure 102 Results of the testing datasets stress level Vs steps

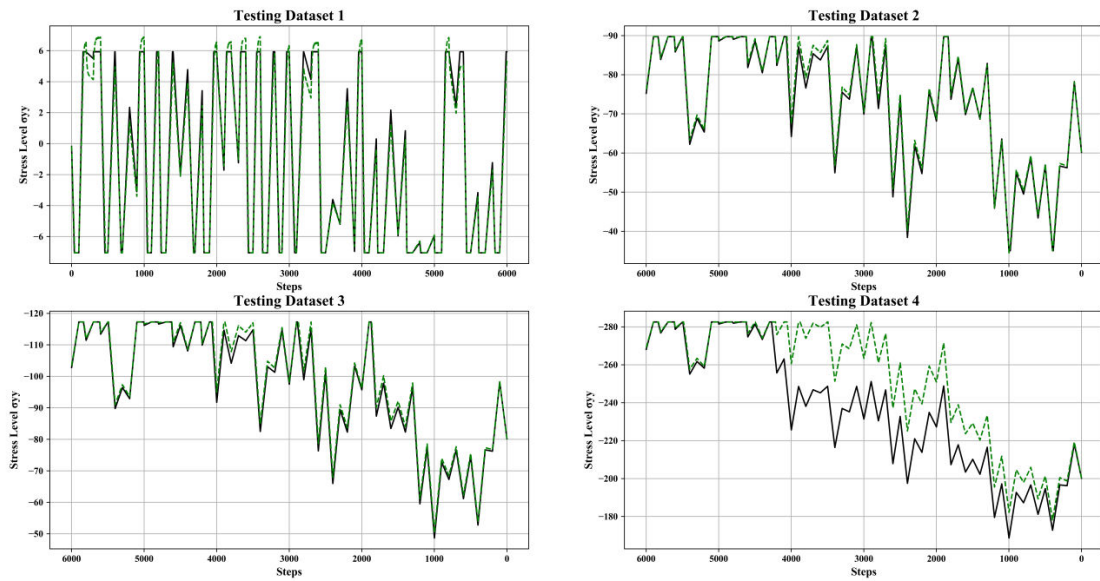


Figure 103 Results of training with the testing Datasets Stress Vs steps

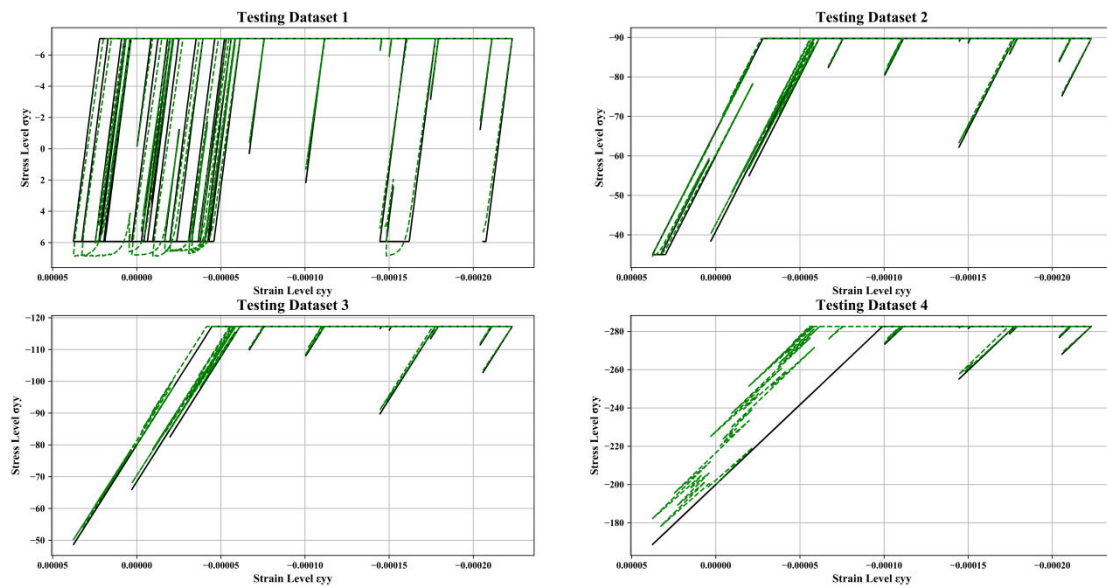


Figure 104 Results of training with the testing Datasets Stress Vs Strain level

#### 4.4.2.3 Testing the Generalisation Ability – Noise in the Training Dataset

The Neural Network’s generalisation ability is also tested with the implementation of noise in the test datasets presented at chapter 4.4.2.2. The idea behind this is that noise can provide a first estimate on how the Neural Network would react when tested with noisy laboratory or field data. Thus, adding noise to the dataset will give a first estimation of how the network would perform with laboratory data (Morales, Luengo, Garcia, Lorena, de Carvalho, & Herrera, 2017).

The noise will be created from a Gaussian distribution with a mean of 0 and a standard deviation of 1 (Figure 105). From Figure 105 it can be observed that the noise level is somewhat high in this case featuring maximum stress levels of  $3kN/m^2$ . For each dataset a set of random numbers is selected and added to the inputs and outputs. Thus, the final “noisy” dataset will be created as  $noisy\ dataset = starting\ dataset + noise$ . (Box, Jenkins, Reinsel, & Ljung, 2016). After noise is added to them, the datasets are tested with the Neural Network. The outcome can be observed in Figure 106 and Figure 107. From the figures it can be concluded that the noise will not affect the final prediction when the Neural Network is trained by using non-noisy data. A comparison with the errors of the normal datasets (calculated at chapter 4.4.2.1) can be seen in Table 21. As it is expected the errors of the noisy datasets are magnified. However, they are still between logical bounds which prove the generalisation ability of the network.



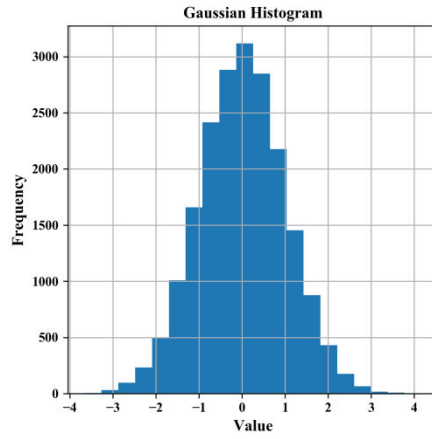


Figure 105 The distribution of the Noise Numbers

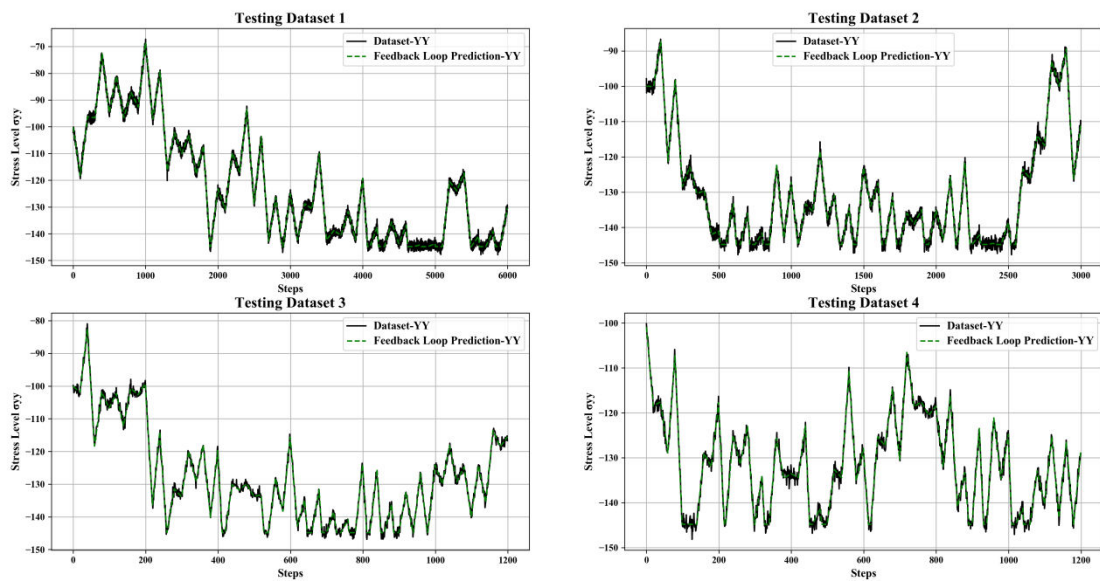


Figure 106 Noisy Dataset testing results stress level Vs steps

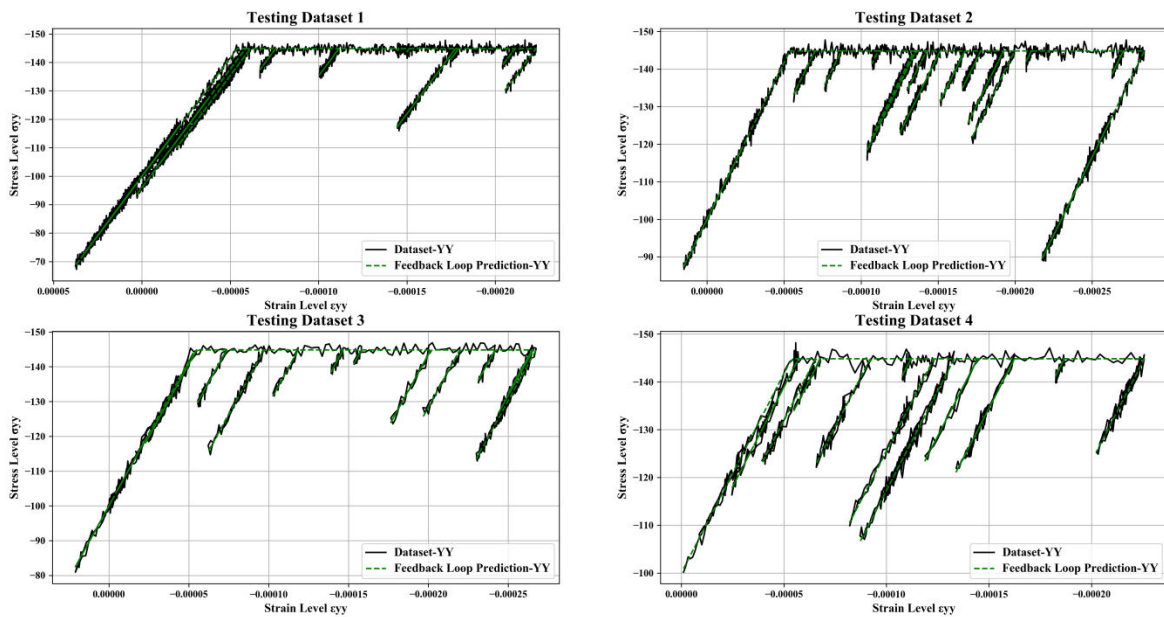


Figure 107 Noisy Dataset testing results stress level Vs strain level

Table 21 Results of different testing dataset with and without noise

	Steps in PLAXIS	Mean Absolute Error in the y direction	
		Without Noise	With Noise
Training Dataset	100	0.017	0.017
Testing Dataset 1	100	0.044	0.807
Testing Dataset 2	50	0.007	0.789
Testing Dataset 3	20	0.014	0.828
Testing Dataset 4	Random Uniform from 1 to 100	0.031	0.797

The final step in this process is to determine how the Neural Network performs when noise is added to the training dataset. A successful training in this case is defined as one where the Neural Network will not take into account the noise as a pattern of the dataset. Thus, the resulting Neural Network should be similar to the one produced in chapter 4.4.1. If this process is successful then the network can be trained by using noisy laboratory data without processing them to reduce the noise before attempting training.

Firstly, the noise has to be determined. The noise is produced by a Gaussian distribution with a mean of 0. The effect of noise in the data will be tested by training the Neural Network with different noise levels. The noise level is determined by the standard deviation of the Gaussian distribution. The effects of noise can be observed in Figure 108. It can be seen that the errors reduced significantly with a standard deviations smaller than 0.2. This is a reasonable noise for the stress levels in the dataset. In Figure 109 and Figure 110 the training dataset can be observed after the addition of noise.

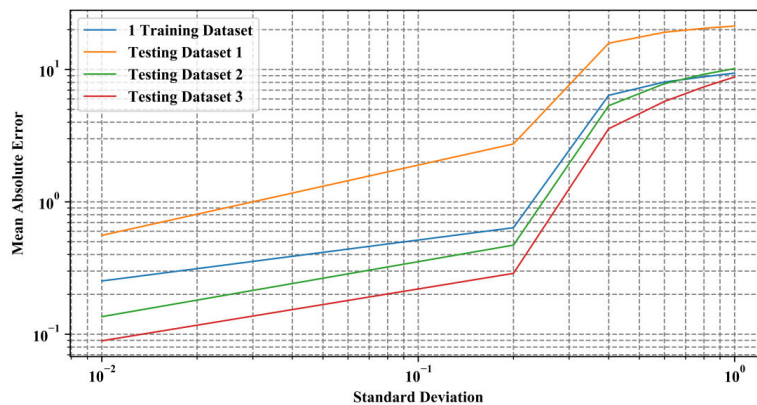


Figure 108 Trained Neural Network error results with different noise levels

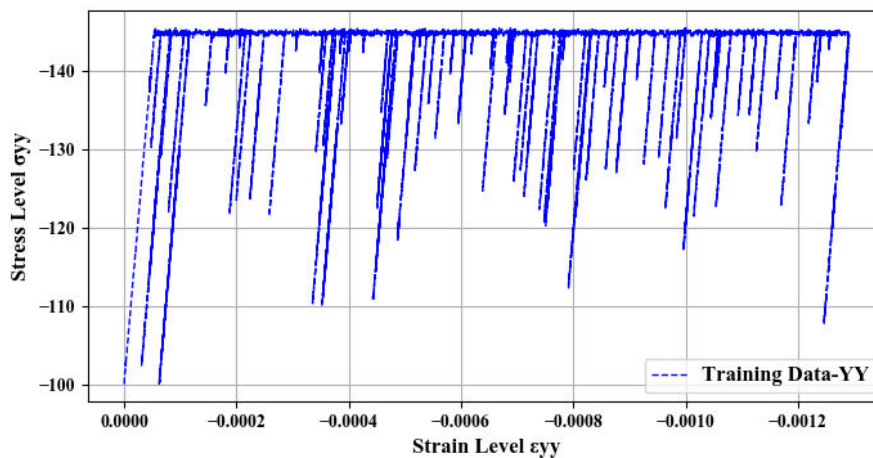


Figure 109 The training dataset with added noise

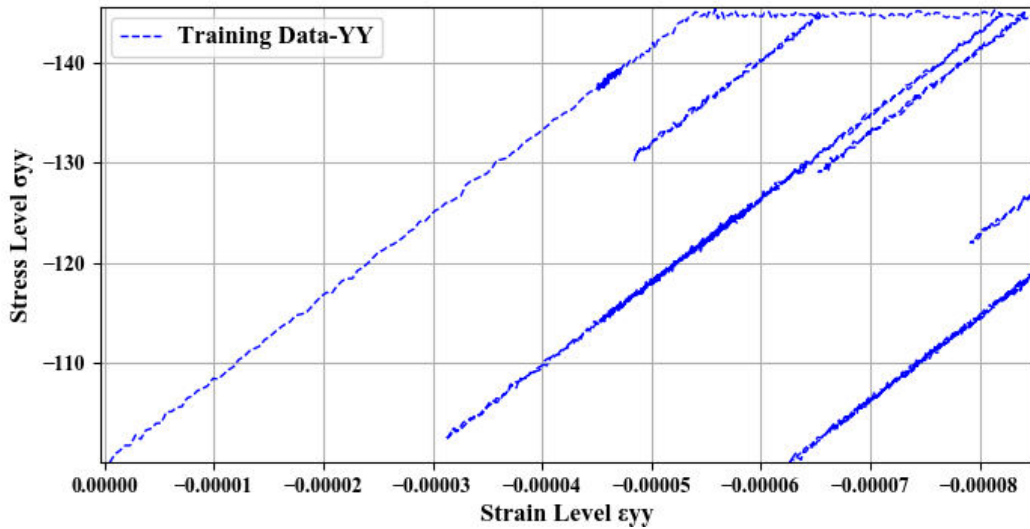


Figure 110 The training dataset with added noise. Zoomed

According to the results it can be observed that the Neural Network has picked up noise as a part of the training dataset (Figure 111). This makes the Feedback prediction challenging and in the end only the yield surface is predicted correctly (Figure 112). However, it can be seen that the results of the feedback prediction although erroneous produces a smooth prediction (Figure 113). Finally, the results of the testing dataset are observed (Figure 114 and Figure 115). It is obvious that the noise training will result in a poorer generalisation than in the chapter 4.4.1, as the errors are larger.

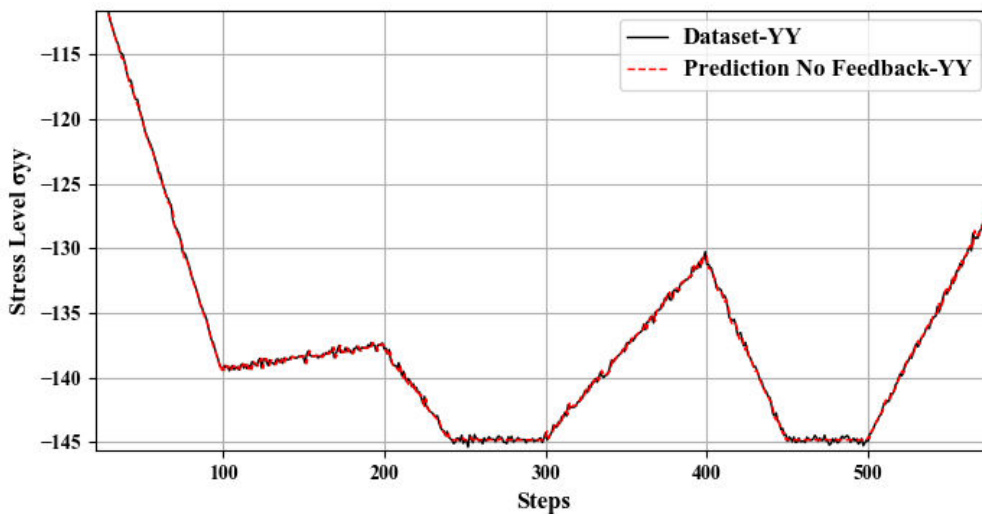


Figure 111 Trained Neural Network results zoomed. The training has pick up the noise

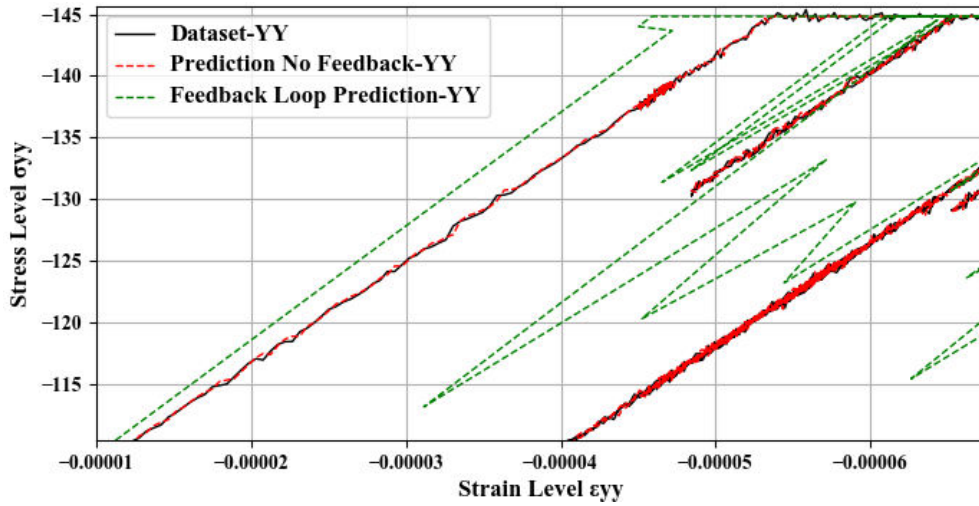


Figure 112 Trained Neural Network with noise stress level Vs Strain level.

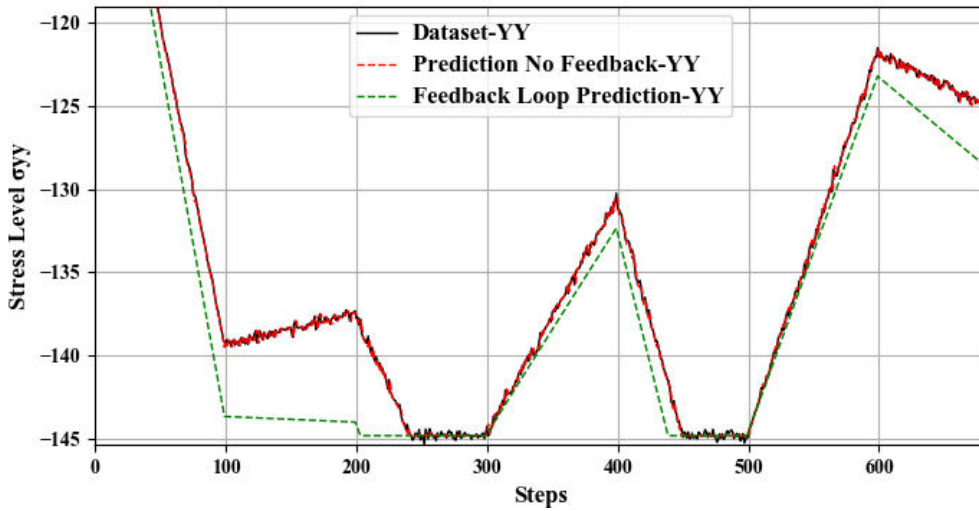


Figure 113 Trained Neural Network results zoomed. Feedback also depicted in the figure

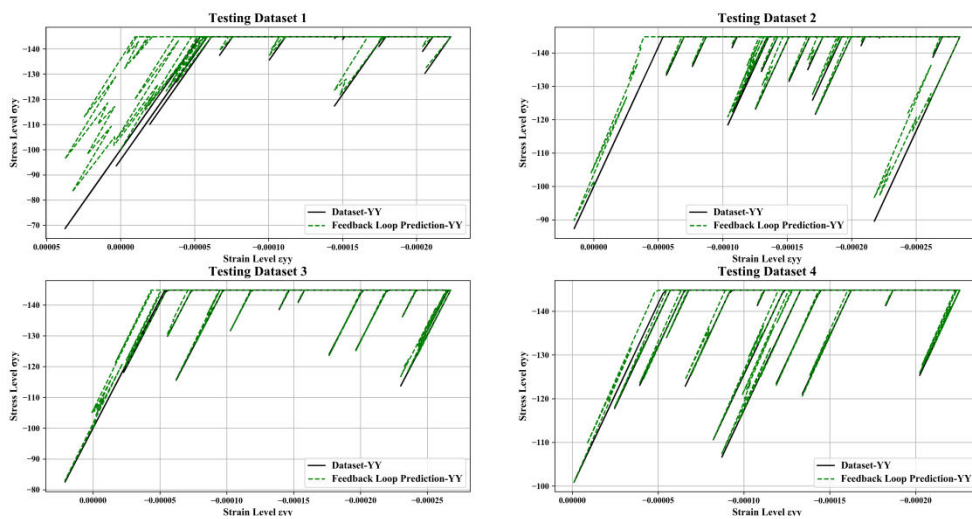


Figure 114 The results of testing stress level Vs strain level

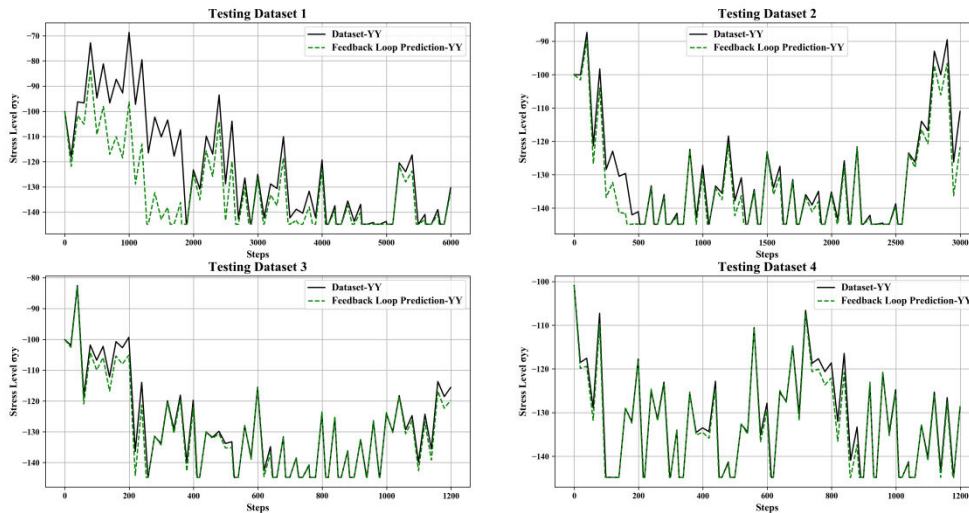


Figure 115 The results of testing stress level Vs steps

## 4.5 Linear-Elastic Work hardening Model

### 4.5.1 Defining Network Architecture and Training Results

As it was done in chapter 4.4.1 the first step in the process is to define a successful architecture for the Feedback prediction. Several numbers of nodes and layers were tested to find the optimum architecture; the activation function chosen is the ReLU function. In Figure 116 the results can be observed, in this case the mean absolute errors of the Feedback prediction are calculated. The mean absolute errors are calculated as in equation 16. There are several architectures with low errors, for example the 4 layered 9 noded one. However, the most convenient in this case is the 1 layer with 6 noded one because it will be more efficient to train.

The results of the Neural Network after training can be observed at Figure 117 and Figure 118. In the Figure 117 it can be observed that the stress unloading reloading behaviour is captured well in the Network. However, the right stress levels are not reached in this case which leads to the Feedback prediction not finally capturing the stress strain relationship (Figure 118). The main issue in this case is that the slope of the yield surface is not captured by the feedback prediction. Therefore, the change from elasticity to plasticity is not captured correctly by the network.

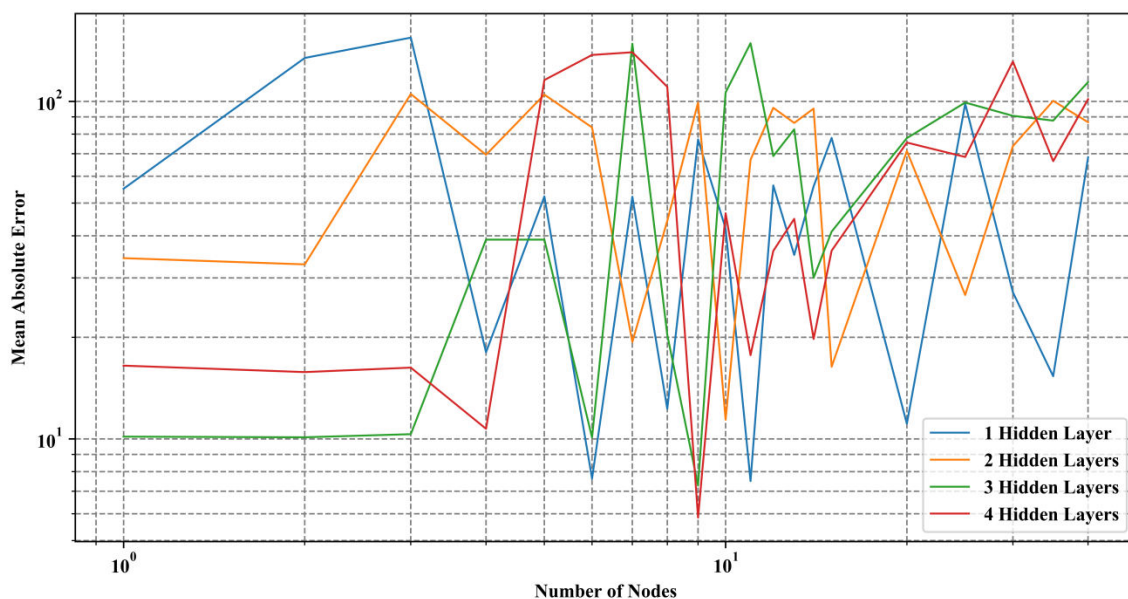


Figure 116 Errors after Training for ReLU activation functions

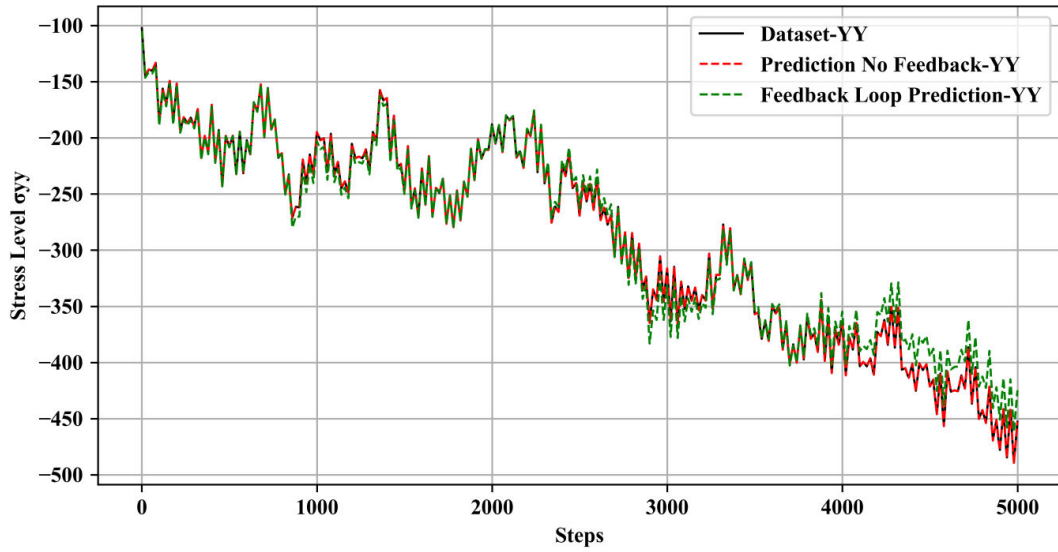


Figure 117 The stress level Vs steps Work Hardening Model

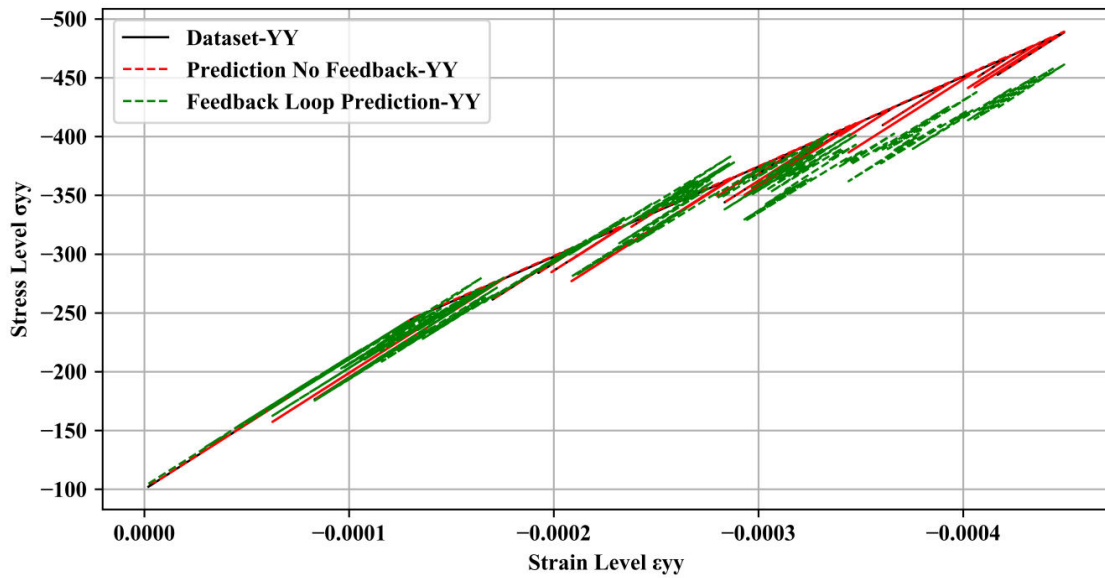


Figure 118 The stress level Vs strain level Work Hardening Model

Nonetheless, the inner workings of the network have to be further analysed. Firstly, the Relative Contribution of each input is evaluated from Table 22. It can be easily concluded that the input of stress has a bigger contribution on the results than the strain increment input. This is further observed in Figure 119 where the connection weights are observed, bigger weights appear with bolder colours in this case and thus the connections of the strain increments appear weaker.

Table 22 The Relative Contribution of each input

RC1 strain input	RC2 stress input
1.26 %	98.74 %

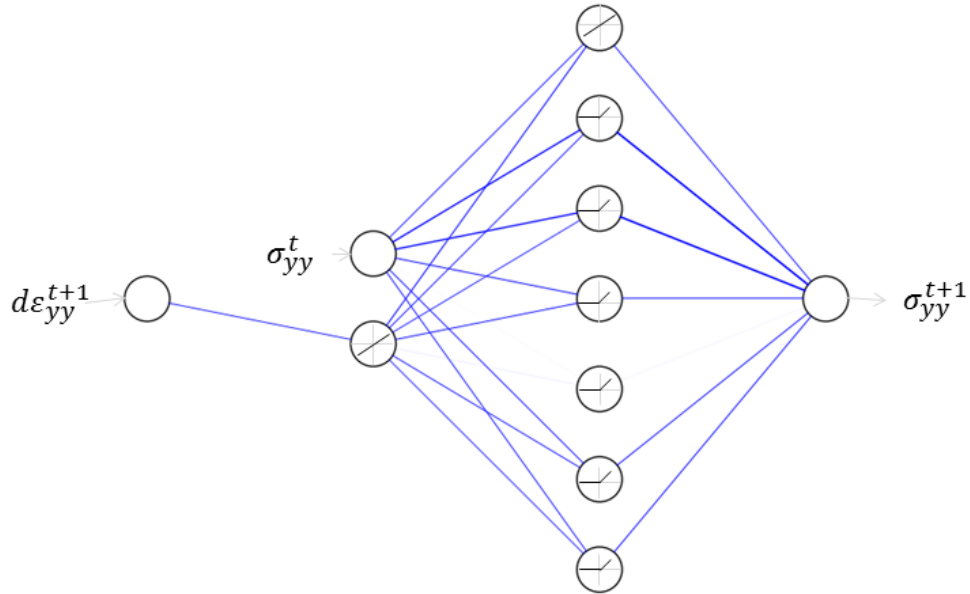


Figure 119 Neural Network Work hardening architecture bolder connections carry bigger weights

Each of the node activations is observed for a loading case during plasticity and a loading case during elasticity (Figure 120). The fact that the activation have no significant difference suggests that the Neural Network is trained based on the stress level rather that identifying the difference between the elastic and the plastic part.

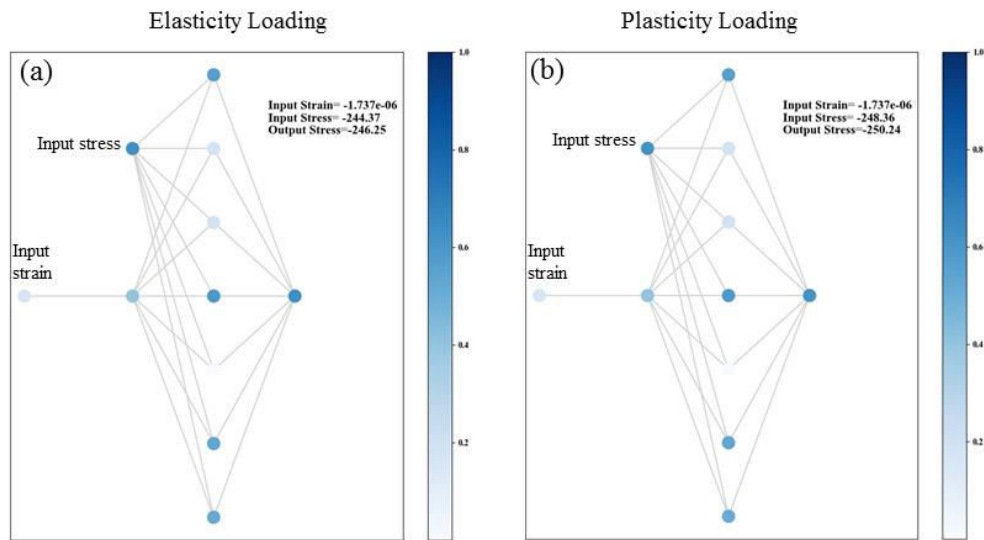


Figure 120 Activations of the Network during (a) elasticity (b) plasticity

## 4.6 Conclusions of Component Based Neural Network Models

After creating the Neural Network soil constitutive models their results were discussed in chapter 4.3. From these results certain conclusions can be made for each of the Neural Networks as a whole.

### 4.6.1 Linear- Elastic Neural Network

The linear elastic model had a straightforward design as a neural network. Linear activation functions were used in the nodes to model the linear relationship between inputs and outputs. The model can easily incorporate more than one direction of stress outputs. The weight connections between the input and hidden layer are sufficient to capture the different gradients describing the relationship of each input with the output. However, when more directions are added to the output then the weight matrix has to be more complex since the influence of each of the inputs has to be taken into account by the Network.

For the results of the Network (chapter 4.3.2) it can be observed that a Network with three hidden nodes can model the 3D stress-strain matrix adequately. Each of the connections corresponds to a certain value of the linear-elastic matrix. From there the parameter estimation problem can be solved and the values of Young's Modulus (E) and Poisson's ratio (v) can be easily retrieved through a simple mathematical operation.

$$\frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu \\ \nu & 1-\nu & \nu \\ \nu & \nu & 1-\nu \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} \end{bmatrix} \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{bmatrix} d\varepsilon_{xx} \\ d\varepsilon_{yy} \\ d\varepsilon_{zz} \end{bmatrix}$$

$$\frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu \\ \nu & 1-\nu & \nu \\ \nu & \nu & 1-\nu \end{bmatrix} = \begin{bmatrix} w_{10}w_1 + w_{13}w_2 + w_{16}w_3 & w_{10}w_4 + w_{13}w_5 + w_{16}w_6 & w_{10}w_7 + w_{13}w_8 + w_{16}w_9 \\ w_{11}w_1 + w_{14}w_2 + w_{17}w_3 & w_{11}w_4 + w_{14}w_5 + w_{17}w_6 & w_{11}w_7 + w_{14}w_8 + w_{17}w_9 \\ w_{12}w_1 + w_{15}w_2 + w_{18}w_3 & w_{12}w_4 + w_{15}w_5 + w_{18}w_6 & w_{12}w_7 + w_{15}w_8 + w_{18}w_9 \end{bmatrix}$$

Thus:  $\frac{E\nu}{(1+\nu)(1-2\nu)} = w_{11}w_1 + w_{14}w_2 + w_{17}w_3 = K1$  [1]

And:  $\frac{E(1-\nu)}{(1+\nu)(1-2\nu)} = w_{10}w_1 + w_{13}w_2 + w_{16}w_3 = K2$  [2]

$$1 \div 2 \leftrightarrow \frac{K1}{K2} = \frac{\nu}{1-\nu} \leftrightarrow \nu = \frac{K1}{K1 + K2} = \frac{576383,2}{(1345253 + 576383,2)} = 0.29$$

And Finally:  $E = \frac{(1+\nu)(1-2\nu)K1}{\nu} = 1076842.82$

The parameters can be easily back calculated in this case which proves that the Neural Network has uncovered the same relationship governing the inputs and the outputs. The Poisson's ratio estimation is really accurate with a percentage error of 3.33% and the Young's modulus has a percentage error of 7.7%. It can also mean that Neural Networks can be a useful tool in the fitting process of data since the process is fast with high accuracy.

Therefore, for all the input and output combinations tested with Neural Networks the results were accurate. The reason behind this is that the nature of the problem is linear and well defined (strain increment inputs, stress increment outputs).

All the datasets tested in chapter 4.3 are linear elastic. However, there are other forms of elasticity which is non-linear. For example hyperelastic material models (W.Humrickhouse, 2009). The question raised for this type of material is if the Neural Network consisting of linear layers would be able to model this type of behaviour. Or will the non-linearity of the constitutive model require non-linear activation functions in the nodes of the network?

#### 4.6.2 Linear-Elastic Perfectly-Plastic

The linear-elastic perfectly-plastic dataset requires more complicated Neural Network architecture. Replicating the continuum mechanics approach the friction element was defined by ReLU activation function. Moreover, the architecture of that Neural Network was defined by the optimum amount of layers and nodes minimising the error after training.

The network is then trained successfully with an architecture consisted of 2 layers with 2 neurons in each layer. From assessing the inner workings of the network the main function of the network is identified. The network extracts the ultimate yield surface level. When the yield surface is reached the ReLU layers will output a zero as their activation. The final output will be the yield stress after summing all the layer outputs which have zero as their activations. The final output will be the yield stress after the output is rescaled.

By testing different initial stresses as testing datasets, it is concluded that the network is not robust enough to capture the full effect of the yield surface. This implies that it cannot be used for another stress path. However, if the network is retrained the yield surface of the network is identified and the stress paths can be followed accurately.

Finally, the "noisy" testing datasets perform well when tested with dataset trained without the use of noise data. The noise is not recognised in this case as a pattern of the data. Nonetheless, the network was trained with a dataset with noise is added to it. In this case the noise is recognised as a pattern for the network and although the simple prediction is accurate the feedback prediction fails to follow. This suggests that this level of noise (taken from a Gaussian distribution with a mean of 0 and a standard deviation of 0.2) is not producing a successful result. Therefore, it is likely that the noise contained in laboratory data will also be recognised as a pattern from the Neural Network.



The drawback of this method can be identified in the different initial stresses. The Network is unable to identify the yield surface. Be that as it may, the network can be expanded to recognise the yield surface by expanding the training dataset to include more stress paths. Therefore, the training dataset should consist of several stress paths and their constitutive parameters.

In addition, the Network was not successful in modelling the yield surface in tension. That is due to the fact that the ReLU accounts only for one ultimate state. Nonetheless, if activation functions, that include two ultimate states, are incorporated to the Neural Network then it will be able to model both the ultimate state of compression and tension. Such activation functions are the Sigmoid and the Tanh because they possess two ultimate state values (Figure 121).

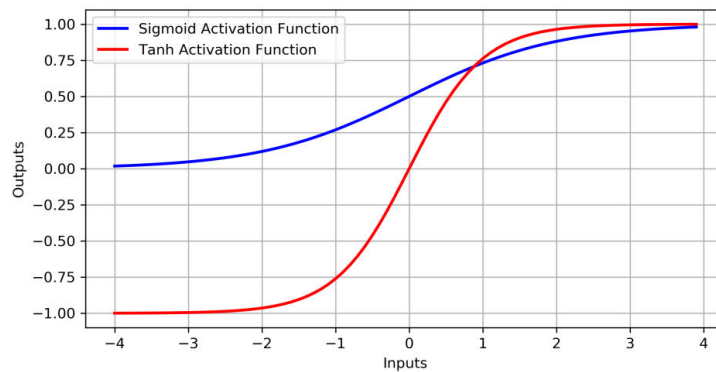


Figure 121 The Activation Functions Sigmoid and Tanh

#### 4.6.3 Linear-Elastic Work-Hardening Model

The hardening model is defined in the same continuum mechanics way as the perfectly-plastic one. The plasticity part needs to be defined by an optimum architecture. After the errors are minimised the training dataset is tested for its accuracy. The feedback prediction is not accurate in this case. The inner activations of the nodes uncover the fact that the plasticity is indistinguishable from elasticity. Thus, the network makes a prediction by only depending on the input of the stress level and does not take into account the input of the strain increments. The reason behind this is the correlation between the inputs and the desired output.

During training the weights are optimised by the gradient of the error between target and predicted outputs (chapter 2.1.4). In that way the weights will be corrected to the right value. However, certain weight connections are more effective in producing the correct output of the network. In this case those connections refer to the input of the stress level. That is the case because the correlation between input stress level and the output stress level is quite strong. In Table 23 the coefficients of correlation between inputs and outputs are shown. It can be seen that the stress input has a coefficient of correlation of 0.99 which implies that there is a strong linear relationship between input and output (Rummel, 1976). On the other hand, the input of the strain increment has a coefficient of correlation of 0.04 which suggests that there is no linear relationship between input and output. Therefore, the network will unavoidably be trained with a major contribution corresponding to the stress level. Thus, this type of issue can be attributed to the way the data were defined as inputs.

Table 23 Coefficient of Correlations between inputs and output

	Coefficient of Correlation with the Stress Output
Input Stress Level	0.99
Input Strain Increment	0.04

The next step is to consider possible solutions of this problem. The first one and quite simple solution is to redefine the Neural Networks inputs and outputs. It can be calculated that the coefficient of correlation between the strain level as an input and the stress level as an output is 0.99. Thus, those two values have a strong linear relationship. The fact that both of them have this strong connection may resolve the issue of the partial contribution of the inputs. If that is not the case then this issue can be resolved by using a Recurrent Neural Network. If a Recurrent Network is used then the errors will be minimized to fit the feedback prediction rather than the normal one.

# Chapter 5. Discussion

---

In this part of the thesis an attempt is made to interpret and describe the significance of the findings in light of what was discussed in the literature study. The aim of the thesis, as it is stated in the Introduction is to assess if the creation of a Neural Network Constitutive Model is possible, with the intention of creating a more data-driven approach to constitutive models.

In the following section of the discussion the research questions posed in the Introduction are answered. The first research question posed is if Neural Networks can behave as successful constitutive material models. From the results of the component based model (especially chapters 4.3 and 4.4) and the results of the new input and output setting in the generic model (chapter 3.3.3.1) it can be concluded that Neural Networks can in fact be trained to behave as constitutive models. However, hardening stress-strain relationships proved to be more challenging to model. Nonetheless, changing the inputs and outputs seems to be promising and is the most accurate solution of this study for the work-hardening plasticity.

The second scientific question concerns the ability of Neural Networks to model typical soil behaviour, for example elasticity or plasticity. Using component based modelling; plasticity and elasticity were modelled successfully with Neural Networks. The loading-unloading behaviour was also captured by the Neural Network material models. The hardening behaviour, however, was difficult to model and required a reassessment of inputs and outputs. However, it was partially captured when new inputs and outputs were assigned to the model (chapter 3.3.3.1).

The third and final question is if laboratory data can be used to train the Neural Network and produce a successful constitutive model. Laboratory data were not used for training in this study, because their non-linear and three-dimensional behaviour will not be captured with the Neural Networks created in this study. Nonetheless, noise was added to the dataset to observe the network's response to laboratory data. The "noisy" dataset is used to train the Neural Network. After training, the Neural Network identified noise as a pattern of behaviour in the dataset. This implies that training with laboratory data will result in uncovering a wrong constitutive relationship. Therefore, training with laboratory data requires some pre-processing to eliminate the noise.

After answering the scientific questions posed in the introduction, major findings of the thesis project are discussed. The first major finding is that the generic Neural Network was unsuccessful in modelling the stress-strain relationship. This is attributed to the varying contributions of the inputs to the output. This statement can be generalised and it can be concluded that if the Network has multiple inputs then the weights will be adjusted to put emphasis on the input that possesses the largest correlation with the output. Throughout the thesis there is a strong linear connection between the stress input and stress output, but there is no linear connection between strain input and stress output, something that was not captured by the Neural Network. In Figure 122 the results of the feedback prediction on the increment of stress can be observed. It is seen that, for the Network's prediction, the increment of stress follows a certain linear relationship that does not distinguish elastic from plastic behaviour. Therefore, the Generic Network (chapter 3.3.2.2) does not distinguish elasticity from plasticity. Constitutive models possess functions that are able to distinguish plastic from elastic behaviour. Thus, the behaviour of the Mohr Coulomb model is not captured correctly by the Network. This is further evaluated in the sensitivity study in chapter 3.3.2.2. Finally, it was observed that this type of behaviour was not documented in the literature study. Thus, it is significant to note that attempts similar to the generic Neural Network will not be successful due to the high linear relationship between stress input and output.

The Network's inputs and outputs were then altered to  $d\varepsilon_{yy}^{t+1}$ ,  $\varepsilon_{yy}^t$ ,  $\sigma_{yy}^t$ , that are the inputs, and  $d\sigma_{yy}^{t+1}$ , which is the output. The results of the feedback and the normal prediction in this case are almost the same (Figure 45 and Figure 46). In Table 24 the coefficient of correlation is presented together with the relative contribution for all the inputs. The relative contribution seems to have an impact on the final Neural Network stress-strain relationship output. The input with the smallest coefficient of correlation has the smallest relative contribution to the network and vice-versa. The results can be observed in Figure 123, in the graph the plastic and elastic components are distinguished from each other as two different linear relationships. This implies that the Network performs closer to a real constitutive model. The stress increments can be divided into plastic and elastic ones. Finally, this network is more successful during the feedback prediction as the increment of strain plays an important role in determining the output. Although it is not a completely successful model it does lead to important conclusions and is the most promising in this case. However, it suggests that a redefinition of input

and outputs is required (Ghaboussi, Pecknold, & Zhang, 1998) or even more complex ways of training the Neural Network (Gandomi & Yun, 2014; Unger & Könke, 2009).

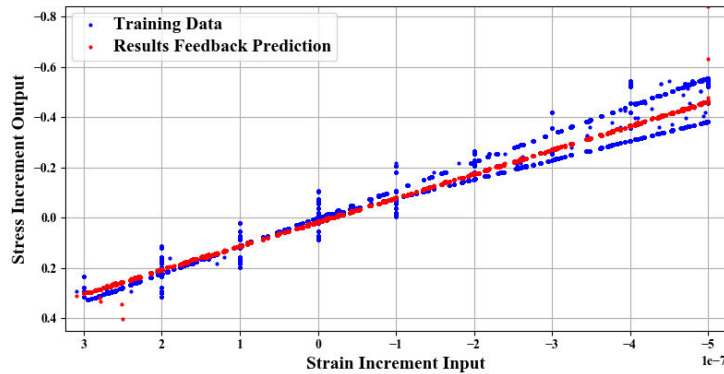


Figure 122 Results of the Generic Neural Network

Table 24 Coefficient of Correlation and Relative Contribution

	Strain Increment Input	Stress Level Input	Strain Level Input
Coefficient of Correlation	0.98	0.007	0.004
Relative Contribution (%)	64	29.4	6.6

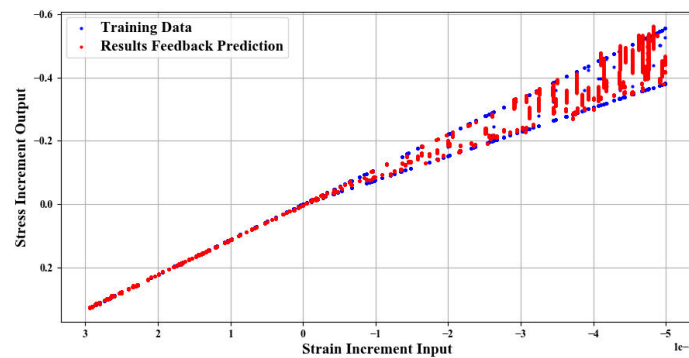


Figure 123 Results of Generic Neural Network with New Inputs and Outputs

In the case of the three-dimensional input the linear-elastic Neural Network model (chapter 4.3.2) has the ability to reproduce the full linear-elastic matrix. From there it is a straightforward process to correlate the soil parameters with the weights of the network. This suggests that parameters can be retrieved from a successfully trained Neural Network. In addition, the linear-elastic model transitions easily from one to three directions of inputs with the addition of proportional number of nodes to the Network (one linear node to one direction of input and 3 linear nodes to three directions of input). This suggests that multiple directions of inputs can be modelled with proportionally wider Neural Network layers.

Moreover, the successful linear-elastic perfectly-plastic Neural Network was trained for only a certain initial stress. The Network was accurate in calculating the stress increment strain increment relationship which is seen in Figure 124. Therefore, the unloading-reloading behaviour is captured accurately. In addition, the stress-strain relationship with the output of stress successful as there is a clear divide between plasticity and elasticity. When different initial stresses are chosen then the final ultimate stress level will not be the same as the training dataset's. In the case of different initial stresses the Neural Network should be retrained to adjust to the new ultimate stress level. However, the training dataset can also be expanded to contain data from different initial stresses with an initial stress as an extra input of the network.

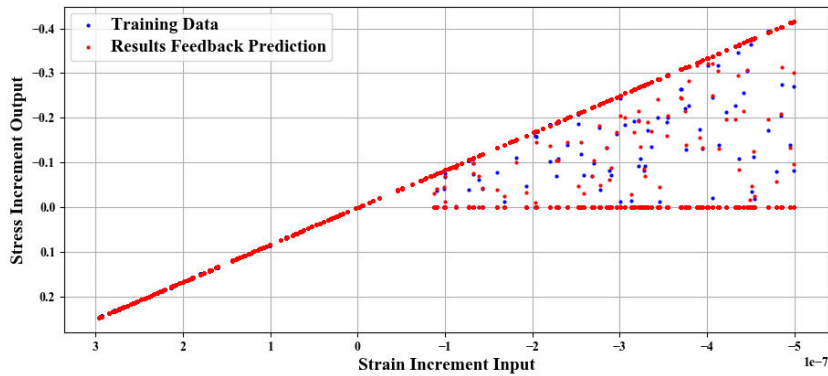


Figure 124 Results Component Based Model LEPP

In chapter 4.4.2.2 and specifically in page 71 the Neural Network of linear-elastic perfectly-plastic was successful in training with the testing data from different stress paths. These datasets consisted of 6000 stress-strain points. These datasets are 98% smaller than the training datasets. This suggests that the training dataset can consist of 1/41 of the data and still be successful. Circling back to the statement made in the literature study (chapter 2.3.1.1), this proves that the training set should be representative of the behaviour that the data are attempting to model (Maters, 1993).

The main function of a soil material model is to be part of the Finite Element Analysis (FEA). In the FEA the material models perform two specific functions. Firstly, they are used to update stress states given the current stress-strain and the strain increment. In this case no integration scheme is required in the FEA when a Neural Network is used for predictions. The Neural Network will automatically provide an updated state of stress for an updated state of strain or strain increment. Numerically this process has the advantage that it eliminates the errors associated with the use of a specific numerical integration scheme.

Secondly, they are used to calculate the material stiffness matrix for the constitutive relation (constitutive matrix or Jacobian). Numerous approaches are suggested to deal with the lack of closed-form expressions for the consistent tangent matrix (Wu, 1991; Zhang, 1996). The most effective one is proposed by Hashash (2004). In this study it is proposed that the stiffness matrix is calculated from the Neural Network. Since the Neural Network provides a relationship between stresses and strains there is an implied stiffness matrix which can be extracted from this relationship (this is also explained in chapter 2.5.2). As noted in the study (Hashash, Jung, & Ghaboussi, 2004) “the use of the matrix is expected to lead to efficient convergence of the Neural integration in the FE analysis”. In that way, the Neural Networks created can be integrated into a FEA if the relationship between stress and strains is outputted as a stiffness matrix.

In addition, Chapter 2.6 of the literature study aims to report applications of Neural Networks in FEA. The models created in this thesis can be part of the self-learning finite element code of H.S. Shin (2000) because the only requirement in this case is that the network predicts stresses based on strains. On the other hand, the “Intelligent Finite Element Method” (Javadi & Tan, 2006) requires the inverse solution of strain plus an output of the elastic modulus. Furthermore, the input is referring in this case to the 3 principal stresses. Thus, the Neural Networks need to be expanded to fit these two specific applications existing in literature.

It is also important to identify limitations of the study in this point. These limitations are significant as they reveal where future research could be applied. A list of them is attached below:

1. The generic and work hardening Neural Networks did not produce successful results.
2. All the Networks apart from the linear elastic Neural Network have only one direction as output of the network.
3. The linear-elastic perfectly-plastic model does not account for tension and different stress levels.
4. The minimum amount of training samples was not investigated in this study.

# Chapter 6. Conclusions

The goal of this thesis project was to determine whether Neural Networks can model constitutive soil behaviour. Over the course of the thesis the goal became clearer as the methodology of each chapter defined the way that the Neural Networks would be formulated. The scope of the project also became narrower in each chapter. This can be easily observed from the difference between the goals of Chapter 3 and Chapter 4. The first approach was to create a generic Neural Network by focussing mainly on how Neural Networks can help uncover the stress-strain behaviour of a constitutive soil model. In Chapter 4 the scope becomes narrower as the Neural Networks aim to model certain soil behaviour like elasticity or plasticity by a component based approach. These network components can be linked together to create successful Neural Network constitutive models.

The main argument of the thesis is that Neural Networks can be used as a substitute for soil constitutive models. The intention behind the main argument is that constitutive models are becoming more parameter depended featuring sometimes more than ten soil parameters. This makes the need for data-driven constitutive models essential. It can be concluded that constitutive models can be created with Neural Networks and those can be accurate and successful as data driven material models, due to the fact that the inputs and the outputs of the Network are only consisted of stresses and strains.

Hardening plasticity was not successfully modelled in this thesis. The reason behind this is that Networks become sensitive towards one of the inputs. However, this sensitivity can be minimised through the use of pre-processing techniques for the inputs, for example fuzzy rules (Jiang, Mahadevan, & Yuan, 2016).

Moreover, component based Neural Networks could be a solution to the “black box” problem of the Neural Networks. If so these components can be used in the same way as the typical components of constitutive model. Thus, they can be arranged to fit any type of soil behaviour. The difference between constitutive models and Neural Networks, as they are defined in this study, is that Neural Networks do not require the input of soil parameters but only the stress-strain data.

In addition, the effect that data have in the training of the Neural Network is prominent throughout the thesis. Especially in Chapter 3 it can be seen that the Coefficient of Correlation of the inputs with the output has an effect on the way the weights are adjusted in the Neural Network. This statement can be further examined in Figure 125. In the Figure it can be observed that a low Coefficient of Correlation will result in a low Relative Contribution and high Mean Absolute Errors. Therefore, it can be concluded that the network will be trained towards the solution with the largest coefficient of correlation, which might not correspond to the solution that the Neural Network intended to identify.

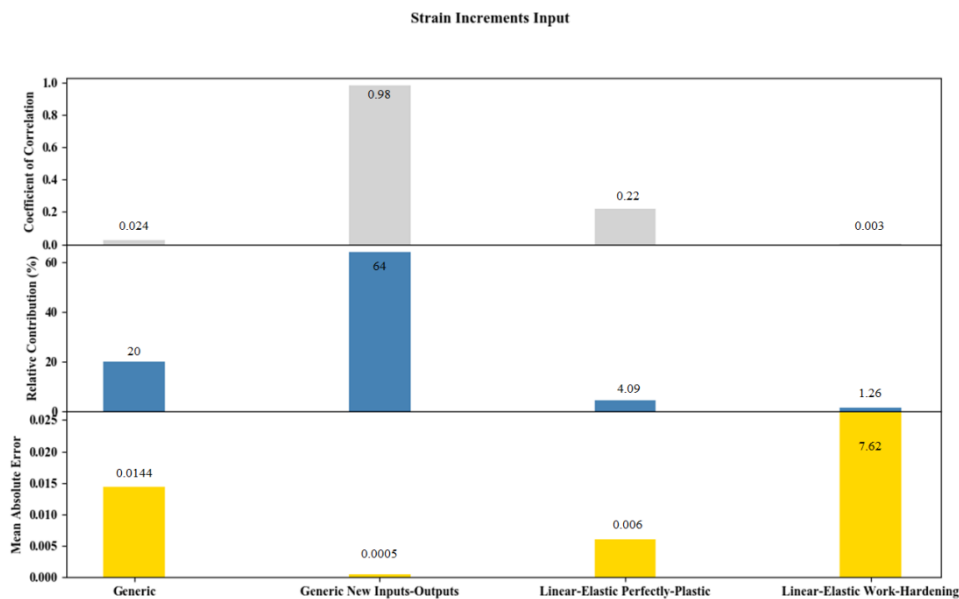


Figure 125 Summary of results for Neural Network Constitutive Models of the thesis

Finally, as a closing statement of the conclusions is important to classify the Neural Networks as successful or not successful and state for each one why that is the case. That can be summarised in Table 25 where all the Networks created in this thesis are posted together with a characterisation of whether they are successful or not and the reason why they are characterised as such. Finally, the chapters where each of these applications can be found are posted as well.

*Table 25 Summary of all the Neural Networks in the thesis*

<b>Neural Network</b>	<b>Is the Network Successful?</b>	<b>Reason</b>	<b>Chapter</b>
<b>Generic</b>	Not Successful	<ul style="list-style-type: none"> <li>• Prediction sensitive to stress inputs. Feedback prediction fails due to this sensitivity.</li> </ul>	3.3.2
<b>Generic New inputs and outputs</b>	Partially Successful	<ul style="list-style-type: none"> <li>• Strong connection between the strain input and the stress output.</li> <li>• Stress input has a smaller contribution to the output.</li> </ul>	3.3.3.1
<b>Linear-Elastic</b>	Successful	<ul style="list-style-type: none"> <li>• Simple linear relationship between inputs and outputs.</li> <li>• Activation function the same as relationship modelled.</li> </ul>	4.3
<b>Linear-Elastic Perfectly-Plastic</b>	Successful	<ul style="list-style-type: none"> <li>• Ultimate stress level so plasticity is represented by one number.</li> <li>• Network can become inactive when plasticity is reached.</li> <li>• Activation function the same as the relationship modelled.</li> </ul>	4.4
<b>Linear-Elastic Work-Hardening</b>	Not Successful	<ul style="list-style-type: none"> <li>• Network is sensitive to the stress inputs.</li> <li>• Feedback prediction fails because of the sensitivity.</li> </ul>	4.5

# Chapter 7. Recommendations for Future Research

Although the thesis was able to answer the scientific questions posed in the Introduction, there are still recommendations for future projects concerning Neural Networks and Constitutive Models.

- First of the effect of Recurrent Neural Networks on modelling hardening behaviour should be accessed. By using a Recurrent Neural Network the Network is trained with the feedback prediction. Thus, the cost function minimized in the back propagation algorithm will refer to the feedback prediction error.
- The main issue of the hardening model is that the network did not recognise any differences between elasticity and plasticity. To help the Network identify this behaviour a decision Neural Network can be added to the training framework. The decision Neural Network will use the input of stress level, strain level and strain increment to predict whether the element is in plasticity or elasticity. This initial Network will be trained separately from the Neural Network stress prediction model. The output of the decision Network will be added as an extra input to the work hardening model. In that way the weights during the training will navigate towards a solution concerning whether the soil element is in plastic or elastic region. The suggested framework is shown in Figure 126.

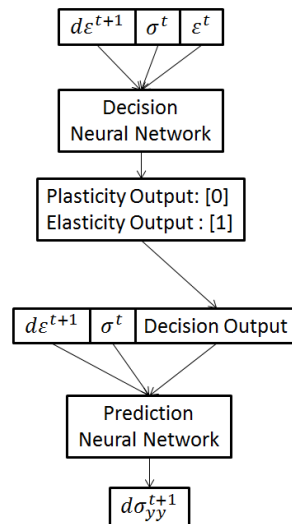


Figure 126 Suggestion of Decision and prediction Neural Network

- The successful Neural Network of perfect-plasticity needs to be expanded to account for different stress paths that will result in various ultimate stresses. That can be achieved if the Neural Network's training dataset is expanded to include more stress paths. In addition to that the initial stresses should also be an input so that the Neural Network can distinguish different stress paths from each other. The Network also needs to be expanded in terms of the stress and strain input and output directions to result in a full direction vector. This can be achieved in the same way it was performed in 4.2.2.1.2. Thus, the amount of neurons needs to be proportional to the amount of stress outputs.
- As it was stated in Chapter 4 the constitutive models are composed out of three main components, the spring, the friction and the dashpot element. The dashpot element was not modelled by a Neural Network in this thesis. Thus, in future projects an attempt should be made to create the dashpot component. The dashpot is used to account for complex soil behaviour like creep and stress relaxation. The equation governing the dashpot element is  $\sigma = \eta \frac{d\epsilon}{dt}$ . Therefore, the Neural Network in this case should have as inputs the stress level and a "time" indication. The non-linearity of a Sigmoid activation function together with the "staging" of multiple layers should be able to model this component.

- The final step in the Neural Network implementation will be to use laboratory or field data to create constitutive models. Here the most concerning element is the quality of the data. If the training dataset is not smoothed then the solution of the Neural Network will incorporate “noise” as a pattern. In this case the effects of noise should be explored thoroughly. Some of the questions that are worth looking into are:
  1. What level of noise will result in the network recognising noise as a pattern?
  2. Are the epochs of training and the architecture of the network enabling in some way the network’s noise identification?

As a general research direction it is important to focus the investigation towards the plasticity hardening behaviour. Different input-output combinations should be implemented, as well as, pre-processing of those inputs to help the network drift towards the aimed solution.



# Appendix A

In this appendix the Neural Network outputs of the linear elastic one input three output model are appended this appendix refers to the Network found in chapter 4.3.1.3.

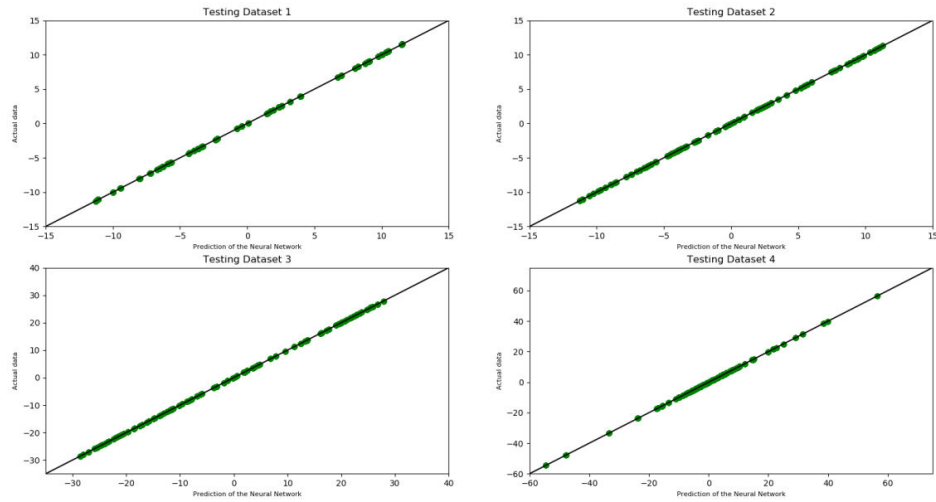


Figure 127 The Testing datasets results of the Z direction

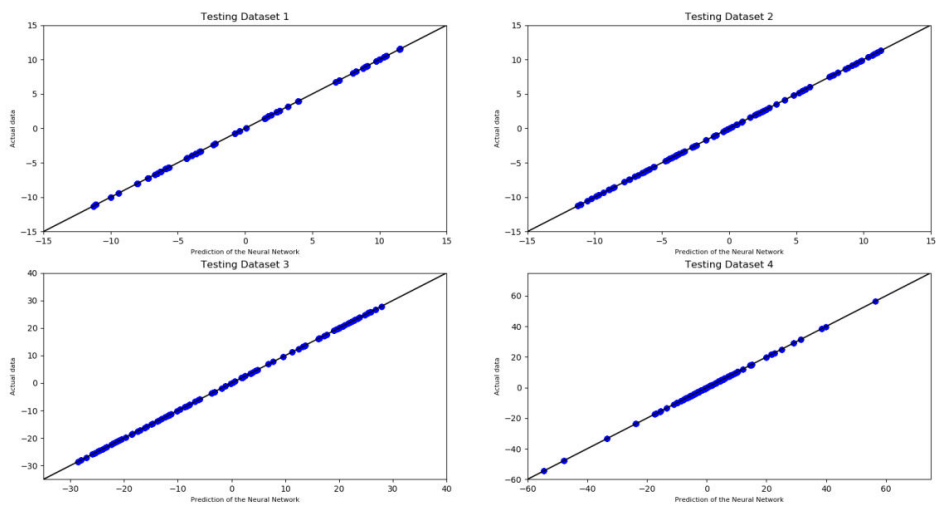


Figure 128 The Testing datasets results of the X direction

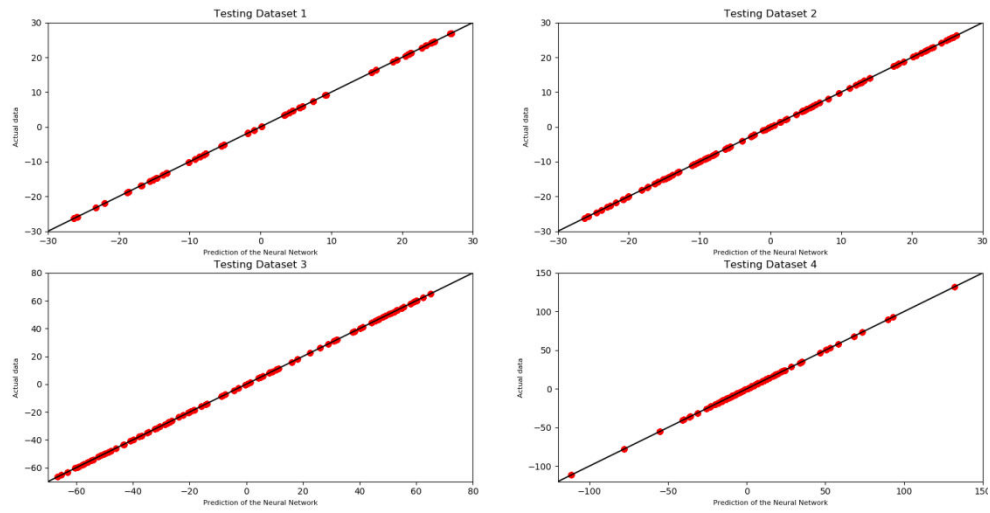


Figure 129 The Testing datasets results of the Y direction

# Appendix B

In this appendix the Relative Contribution metric will be explained. This metric aims to quantify the relationship that has been modelled in the validation of a Neural Network, rather than using an error measure as the only basis for assessment. This will give a first estimation on how the weights of a Neural Network control the interactions between inputs and outputs in the Network. Thus, the contribution of each input can be determined by the strength and direction of the connection weights between them. The Overall connection weight (Olden & Jackson, 2002) is used in this case to determine the contribution of a certain input to the output.

To understand how this metric is calculated an example will be posted in this appendix. By examining Figure 130 the overall connection weight (OCW) of input 1 can be calculated by determining  $c_{A,1}$  and  $c_{B,1}$ . These are the contribution of input 1 through the hidden node A and B. Summing them will result in the OCW as follows:

$$\begin{aligned}
 c_{A,1} &= w_{A,1} \times w_{O,A} \\
 c_{B,1} &= w_{B,1} \times w_{O,B} \\
 OCW_1 &= c_{A,1} + c_{B,1}
 \end{aligned}
 \tag{39}$$

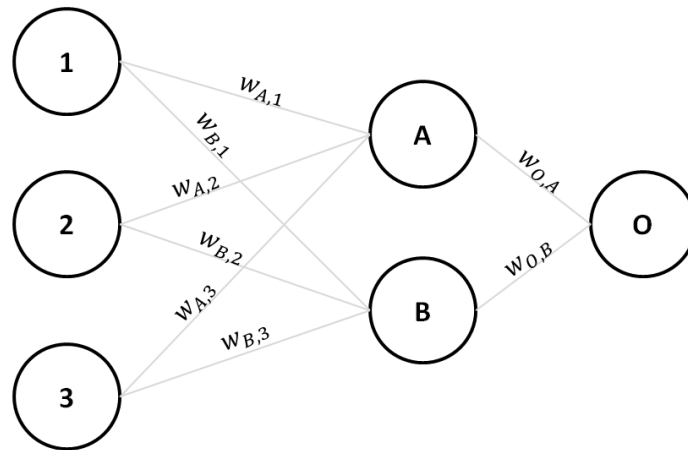


Figure 130 Example Neural Network

It can be observed that the OCW is only an approximation of the real relationship of an input. These metric does not take into account the “reducing” effect activation function like the sigmoid have on the inputs and does not account for the biases. However, Olden et al. (2002) found that the connection weight approach is able to model each input’s importance and identify this as the best methodology in comparison to other commonly used methods. The next step in this process is to define the Relative Contribution (RC) here the contribution of each input is calculated by the percentage of each one to the sum of the absolute values of the OCWs as follows:

$$RC_1 = \frac{OCW_1}{|OCW_1| + |OCW_2| + |OCW_3|}
 \tag{40}$$

# Appendix C

In this chapter the Garson's algorithm will be explained using an example Neural Network. The Network of the example is shown in Figure 131.

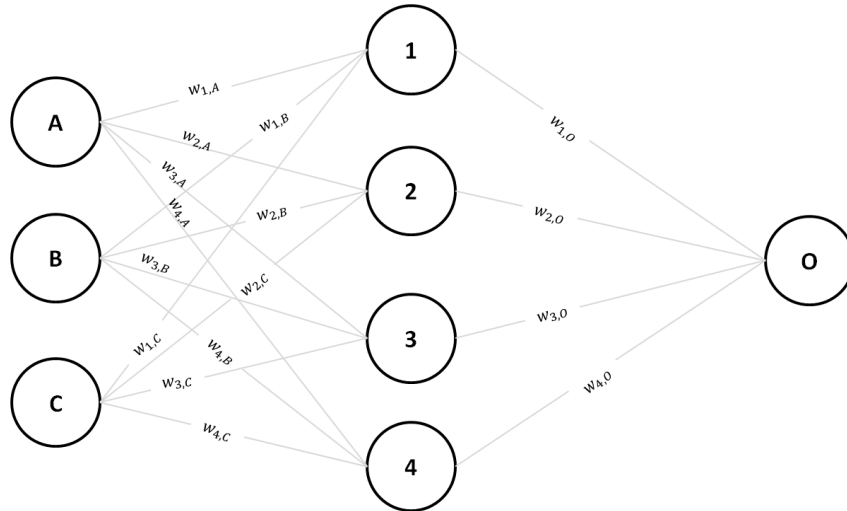


Figure 131 Example Network

The first step in the Garson's algorithm process is to define the weights matrix. In the matrix the weights connecting the neurons between the input- hidden-output layers are shown.

Table 26 Neural Network Weight Matrix

Hidden Neuron	Input A	Input B	Input C	Output
Hidden 1	$w_{A,1}$	$w_{B,1}$	$w_{C,1}$	$w_{O,1}$
Hidden 2	$w_{A,2}$	$w_{B,2}$	$w_{C,2}$	$w_{O,2}$
Hidden 3	$w_{A,3}$	$w_{B,3}$	$w_{C,3}$	$w_{O,3}$
Hidden 4	$w_{A,4}$	$w_{B,4}$	$w_{C,4}$	$w_{O,4}$

Then for each Neuron  $i$ , the absolute value of the hidden-output layer connection weight is multiplied with the absolute value of the hidden –input layer connection weight. This process is further explained in Table 27.

Table 27 Neural Network Connection Calculations

Hidden Neuron	Input A	Input B	Input C
Hidden 1	$P_{1A} =  w_{A,1}  \times  w_{O,1} $	$P_{1B} =  w_{B,1}  \times  w_{O,1} $	$P_{1C} =  w_{C,1}  \times  w_{O,1} $
Hidden 2	$P_{2A} =  w_{A,2}  \times  w_{O,2} $	$P_{2B} =  w_{B,2}  \times  w_{O,2} $	$P_{2C} =  w_{C,2}  \times  w_{O,2} $
Hidden 3	$P_{3A} =  w_{A,3}  \times  w_{O,3} $	$P_{3B} =  w_{B,3}  \times  w_{O,3} $	$P_{3C} =  w_{C,3}  \times  w_{O,3} $
Hidden 4	$P_{4A} =  w_{A,4}  \times  w_{O,4} $	$P_{4B} =  w_{B,4}  \times  w_{O,4} $	$P_{4C} =  w_{C,4}  \times  w_{O,4} $

For each of the hidden neurons of the Neural Network the partition calculated in Table 27 will be divided by the sum for the entire input variable to obtain the  $Q_{ij}$ . For example,  $Q_{ij} = P_{1A} / (P_{1A} + P_{1B} + P_{1C})$ . Then these outputs are summed for each of the inputs. Finally, the  $S_j$  will be divided by the sum of all the input variables. This will be expressed as a percentage which finally gives the relative importance or distribution of all the output weights attributable to a given input variable.

Finally, it is important to note that the Garson's algorithm is only applied to one layered Neural Networks.

Table 28 Neural Network Connection Calculations

Hidden Neuron	Input A	Input B	Input C
Hidden 1	$Q_{1A} = \frac{P_{1A}}{(P_{1A} + P_{1B} + P_{1C})}$	$Q_{1B} = \frac{P_{1B}}{(P_{1A} + P_{1B} + P_{1C})}$	$Q_{1C} = \frac{P_{1C}}{(P_{1A} + P_{1B} + P_{1C})}$
Hidden 2	$Q_{2A} = \frac{P_{2A}}{(P_{2A} + P_{2B} + P_{2C})}$	$Q_{2B} = \frac{P_{2B}}{(P_{2A} + P_{2B} + P_{2C})}$	$Q_{2C} = \frac{P_{2C}}{(P_{2A} + P_{2B} + P_{2C})}$
Hidden 3	$Q_{3A} = \frac{P_{3A}}{(P_{3A} + P_{3B} + P_{3C})}$	$Q_{3B} = \frac{P_{3B}}{(P_{3A} + P_{3B} + P_{3C})}$	$Q_{3C} = \frac{P_{3C}}{(P_{3A} + P_{3B} + P_{3C})}$
Hidden 4	$Q_{4A} = \frac{P_{4A}}{(P_{4A} + P_{4B} + P_{4C})}$	$Q_{4B} = \frac{P_{4B}}{(P_{4A} + P_{4B} + P_{4C})}$	$Q_{4C} = \frac{P_{4C}}{(P_{4A} + P_{4B} + P_{4C})}$
Sum	$S_A = Q_{1A} + Q_{2A} + Q_{3A} + Q_{4A}$	$S_B = Q_{1B} + Q_{2B} + Q_{3B} + Q_{4B}$	$S_C = Q_{1C} + Q_{2C} + Q_{3C} + Q_{4C}$

Table 29 Relative Importance Calculation

	Input A	Input B	Input C
Relative Importance (%)	$RI_A = \frac{S_A}{S_A + S_B + S_C}$	$RI_B = \frac{S_B}{S_A + S_B + S_C}$	$RI_C = \frac{S_C}{S_A + S_B + S_C}$

# Bibliography

---

- (2015). *PLAXIS 2D 2015, Manual*. Plaxis Bv.
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., & Citro, C. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.
- Amari, S., Murata, N., Muller, K., Finke, M., & Hua Yang, H. (1997). Asymptotic Statistical Theory of Overtraining and Cross-Validation. *NEURAL NETWORKS, VOL. 8*, 985-995.
- Bamdad, A., & Habibagahi, G. (2003). A neural network framework for mechanical behavior of unsaturated soils. *Can. Geotech. J.* 40, 684–693.
- Banimahd, M., Yasrobi, S., & Woodward, P. (2005). Artificial neural network for stress–strain behavior of sandy soils: Knowledge based verification. *Computers and Geotechnics* 32, 377–386.
- Bauer, E. (1996). Calibration of a comprehensive hypoplastic model for granular materials. *Soils and Foundations*, 36(1), 13-36.
- Baziar, M., & Jafarian, Y. (2007). Assessment of liquefaction triggering using strain energy concept and ANN model: Capacity Energy. *Soil Dynamics and Earthquake Engineering* 27, 1056–1072.
- Beatty, M., & Byrne, P. (1998). An effective stress model for predicting liquefaction. *Geotechnical Earthquake Engineering and Soil Dynamics III. ASCE Geo-technical Special Publication* 75, 766-777.
- Benitez, J. M., Castro, J. L., & Requena, I. (1997). Are Artificial Neural Networks Black Boxes? *IEEE Transactions on Neural Networks*, VOL. 8, 1156-1164.
- Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time Series Analysis: Forecasting and Control (Wiley Series in Probability and Statistics)*. New Jersey: Wiley.
- Brownlee, J. (2016, July 14). *8 Inspirational Applications of Deep Learning*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/inspirational-applications-deep-learning/>
- Caudill, M. (1988). Neural networks primer, part III. *AI Expert* 3, 53 - 59.
- Chan, W. T., Chow, Y. K., & Liu, L. F. (1995). Neural Network : An Alternative to Pile Driving Formulas. *Computerd on Geotechnics* 17, 135-156.
- Cho, S. E. (2009). Probabilistic stability analyses of slopes using the ANN-based response surface. *Computers and Geotechnics* 36, 787–797.
- Chollet, F. (2015). Keras.
- Clark, C., & Storkey, A. (2015). Training Deep Convolutional Neural Networks to Play Go. *International Conference on Machine Learning*. Lille , France: JMLR: W&CP volume 37.
- Das, S. K., & Basudhar, P. K. (2006). Undrained lateral load capacity of piles in clay using artificial neural network. *Computers and Geotechnics* 33, 454–459.
- Das, S. K., & Basudhar, P. K. (2008). Prediction of residual friction angle of clays using artificial neural network. *Engineering Geology* 100, 142–145.
- Das, S. K., Biswal, R. K., Sivakugan, N., & Das, B. (2011). Classification of slopes and prediction of factor of safety using differential evolution neural networks. *Environ Earth Sci* 64, 201–210.
- David, O. E., Netanyahu, N. S., & Wolf, L. (2016). DeepChess: End-to-End Deep Neural Network. *International Conference on Artificial Neural Networks (ICANN)*.
- Davies, M. (2002, 06 20). *The Neuron: size comparison*. Retrieved from <https://sites.ualberta.ca/~neuro/OnlineIntro/NeuronStructure.htm>

- Ellis, G. W., Yao, C., Zhao, R., & Penumadu, D. (1995). Stress-Strain Modeling of Sands Using Artificial Neural Networks. *Journal of Geotechnical Engineering* 121.
- Ferentinou, M., & Sakellariou, M. (2007). Computational intelligence tools for the prediction of slope performance. *Computers and Geotechnics* 34, 362–384.
- Filip Obrzud, R. (2009). *Numerical Modeling and Neural Networks to Identify Constitutive Parameters from In Situ Tests*. Lausanne: École Polytechnique Fédérale de Lausanne.
- Gandomi, A., & Yun, G. (2014). Coupled SelfSim and genetic programming for non-linear material constitutive modelling. *Inverse Problems in Science and Engineering*.
- Garson, G. D. (1991). Interpreting neural-network connection weights. *AI Expert* 6, 46 - 51 .
- Ghaboussi, J., & Jamshid, S. (2006). Neural network constitutive model for rate-dependent materials. *Computers & Structures* 84, 955-963.
- Ghaboussi, J., & Sidarta, D. (1998). Constitutive Modeling of Geomaterials from Non-uniform Material Tests. *Computers and Geotechnics, Vol. 22.,* 53-71.
- Ghaboussi, J., Garrett, G., & Wu, X. (1991). Knowledge Based Modeling of Material Behavior with Neural Networks. *Journal of Engineering Mechanics, Vol. 117.*
- Ghaboussi, J., Pecknold, D., & Zhang, M. (1998). Autoprogressive Training of Neural Network Constitutive Models. *International Journal For Numerical Methods in Engineering*, 105D126.
- Goh, A. (1995). Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 143-151.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Hanna, A., Morcou, G., & Helmy, M. (2004). Efficiency of pile groups installed in cohesionless soil using artificial neural networks. *Can. Geotech. J.* 41, 1241–1249.
- Hashash, Y. M., Jung, S., & Ghaboussi, J. (2004). Numerical implementation of a neural network based material model in finite element analysis. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, 989–1005.
- Hern, A. (2016, June 28). *Google says machine learning is the future. So I tried it myself*. Retrieved from theguardian: <https://www.theguardian.com/technology/2016/jun/28/google-says-machine-learning-is-the-future-so-i-tried-it-myself>
- Javadi, A. A., & Tan, T. P. (2006). Neural network for constitutive modelling in finite element analysis.
- Jiang, X., Mahadevan, S., & Yuan, Y. (2016). Fuzzy stochastic neural network model for structural system. *Mechanical Systems and Signal Processing*, 394–411.
- Johari, A., Javadi, A., & Habibagahi, G. (2011). Modelling the mechanical behaviour of unsaturated soils using a genetic algorithm-based neural network. *Computers and Geotechnics* 38, 2-13.
- Jordan, M. (1995). *Why the logistic function? A tutorial discussion on probabilities and neural networks*. MIT Computational Cognitive Science Report 9503.
- Jung, S., & Ghaboussi, J. (2006). Neural Network Constitutive Model for Rate Dependent Materials. *Computers & Structures, v 84,* 955-963.
- Kalinli, A., Acar, M., & Gündüz, Z. (2011). New approaches to determine the ultimate bearing capacity of shallow foundations based on artificial neural networks and ant colony optimization. *Engineering Geology* 117, 29-38.
- Kingston, G. B., Maier, H. R., & Lambert, M. F. (2005). Calibration and validation of neural networks to ensure physically plausible hydrological modeling. *Journal of Hydrology* 314, 158–176.

- Koprinkova, P., & Petrova, M. (1999). Data-scaling problems in neural-network training. *Engineering Applications of Artificial Intelligence* 12, 281-296.
- Kung, G. T., Hsiao, C., Schuster, M., & Juang, C. H. (2007). A neural network approach to estimating deflection of diaphragm. *Computers and Geotechnics* 34, 385–396.
- Kurup, P. U., M.ASCE, & Dudani, N. K. (2002). Neural Networks for Profiling Stress History of Clays from PCPT Data. *Journal of Geotechnical and Geoenvironmental Engineering* 128, 569-579.
- Lee, S., Lee, S., & Kim, Y. (2003). An approach to estimate unsaturated shear strength using artificial neural network and hyperbolic formulation. *Computers and Geotechnics* 30, 489-503.
- Lefik, M. (2013). Some aspects of application of artificial neural network for numerical modeling in civil engineering. *The Journal of Polish Academy of Sciences* 61.
- Lefik, M., & Schrefler, B. (2003). Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Comput. Methods Appl. Mech. Engrg.* 192, 3265–3283.
- Maier, H., & Dandy, G. (2000b). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environ. Modell. Softw.* 15, 101-124.
- Maters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press.
- Morales, P., Luengo, J., Garcia, L. P., Lorena, A. C., de Carvalho, A. C., & Herrera, F. (2017). *Noisy Data in Data Mining*. Retrieved from Soft Computing and Intelligent Information Systems, A University of Granada research group: <http://sci2s.ugr.es/noisydata>
- Morgan, N., & Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: some experiments. In *Advances in neural information processing systems 2* (pp. 630-637). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Najjar, Y. M., & Huang, C. (2007). Simulating the stress–strain behavior of Georgia kaolin via recurrent neuronet approach. *Computers and Geotechnics* 34, 346–361.
- Nilsson, N. (2009). *The Quest for Artificial Intelligence a History of Ideas and Achievements*. Cambridge University Press.
- Novák, V., Perfilieva, I., & Mockor, J. (1999). *Mathematical Principles of Fuzzy Logic*. Kluwer Academic Publishers.
- Olden, J. D., & Jackson, D. A. (2002). Illuminating the ‘black box’: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling* 154, 135-150.
- Padmini, D., Ilamparuthi, K., & Sudheer, K. (2008). Ultimate bearing capacity prediction of shallow foundations on cohesionless soils using neurofuzzy models. *Computers and Geotechnics* 35, 33–46.
- Penumadu, D., & Zhao, R. (1999). Triaxial compression behavior of sand and gravel using artificial neural networks (ANN). *Computers and Geotechnics* 24, 207±230.
- Pernot, S., & Lamarque, C. (1999). Application of neural networks to the modelling of some constitutive laws. *Neural Networks* 12, 371–392.
- Provenzano, P., Ferlisi, S., & Musso, A. (2004). Interpretation of a model footing response through an adaptive neural fuzzy inference system. *Computers and Geotechnics* 31, 251–266.
- Raftery, A., & Kass, R. (1995). Bayes Factors. *Journal of the American Statistical Association*, 773-795.
- Rogers, L., & Dowla, F. (1994). Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling. *Water resource Research*, 457–481.
- Romo, M. P., García, S. R., Mendoza, M. J., & Taboada-Urtuzuástegui, V. (2001). Recurrent and Constructive-Algorithm Networks For Sand Behavior Modeling. *The International Journal of Geomechanics*, 371–387.



- Rummel, R. (1976). *Understanding*. Honolulu: Department of Political Science.
- Samui, P. (2012). Three-Dimensional Site Characterization Model of Bangalore Using Support Vector Machine. *ISRN Soil Science*.
- Sarle, W. S. (1999). *Ill-Conditioning in Neural Networks*. NC, USA: SAS Institute Inc.
- Shahin, M. ,, & Jaksa, M. (2004b). Data division for developing neural networks applied to geotechnical engineering. *Journal of Computing in Civil Engineering*, 18(2), 105-114.
- Shahin, M. (2010). Intelligent computing for modeling axial capacity of pile foundations. *Canadian Geotechnical Journal* 47, 230-243.
- Shahin, M. (2012). Artificial Intelligence in Geotechnical Engineering: Applications, Modeling Aspects, and Future Directions. In X. a. Yang, *Metaheuristics in Water, Geotechnical and Transport Engineering* (pp. 169-194). London, UK: Elsevier Science.
- Shahin, M. (2014). Load–Settlement Modeling of Axially Loaded Drilled Shafts Using CPT-Based Recurrent Neural Networks. *International Journal of Geomechanics* 14.
- Shahin, M. A., Jaksa, M. B., & Maier, H. R. (n.d.). State of the Art of Artificial Neural Networks in Geotechnical Engineering. *The Electronic Journal of Geotechnical Engineering (EJGE)* 8.
- Shahin, M. A., Maier, H. R., & Jaksa, M. (2005). Investigation into the Robustness of Artificial Neural Networks for a Case Study in Civil Engineering. *International Congress on Modeling and Simulation* (pp. 79-83). Melbourne: MODSIM 2005.
- Shahin, M., & Jaksa, B. (2003). *Modelling the pullout capacity of marquee ground anchors*. University of Adelaide.
- Shin, H., & Pande, G. (2000). On self-learning finite element codes based on monitored response of structures. *Computers and Geotechnics* 27, 161-178.
- Sidarta, D., & Ghaboussi, J. (1998). A New Nested Adaptive Neural Networks (NANN) for Constitutive Modeling. *Computers and Geotechnics* 22, 29-52.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1929-1958.
- Stefanos, D., & Gyan. (2015). On Neural Network Constitutive Models for Geomaterials. *Journal of Civil Engineering Research* 2015, 106-113.
- Swirszcz, G., Czarnecki, W. M., & Pascanu, R. (2017). *Local Minima in Training of Neural Networks*. London, UK: DeepMind.
- Turner, M., Clough, R., Martin, H., & Topp, L. (1956). Stiffness and Deflection Analysis of Complex Structures. *Journal of the Aeronautical Sciences* 23, 805-823.
- Unger, J., & Könke, C. (2009). Neural networks as material models within a multiscale approach. *Computers and Structures*, 1177–1186.
- W.Humrickhouse, P. (2009, January). Hyperelastic Models for Granular Material. *Ph.D. thesis*. Fusion Technology Institute University of Wisconsin .
- Wu, X. (1991). Neural network-based material modeling. *Ph.D. Thesis*. Urbana, IL: University of Illinois at Urbana-Champaign.
- Zhang, M. (1996). Neural network material models determined from structural tests. *Ph.D. Thesis*. Urbana, IL: University of Illinois at Urbana-Champaign.
- Zhu, J., Zaman, M., & Anderson, S. (1998). Modeling of soil behavior with a recurrent neural network. *Can. Geotech. J.* 35, 858–872.

Zienkiewicz, O. (1985). Numerical Modelling and Geomechanics. In Z.Bazant, *Mechanics of Geomaterials* (pp. 471-499). Wiley.

