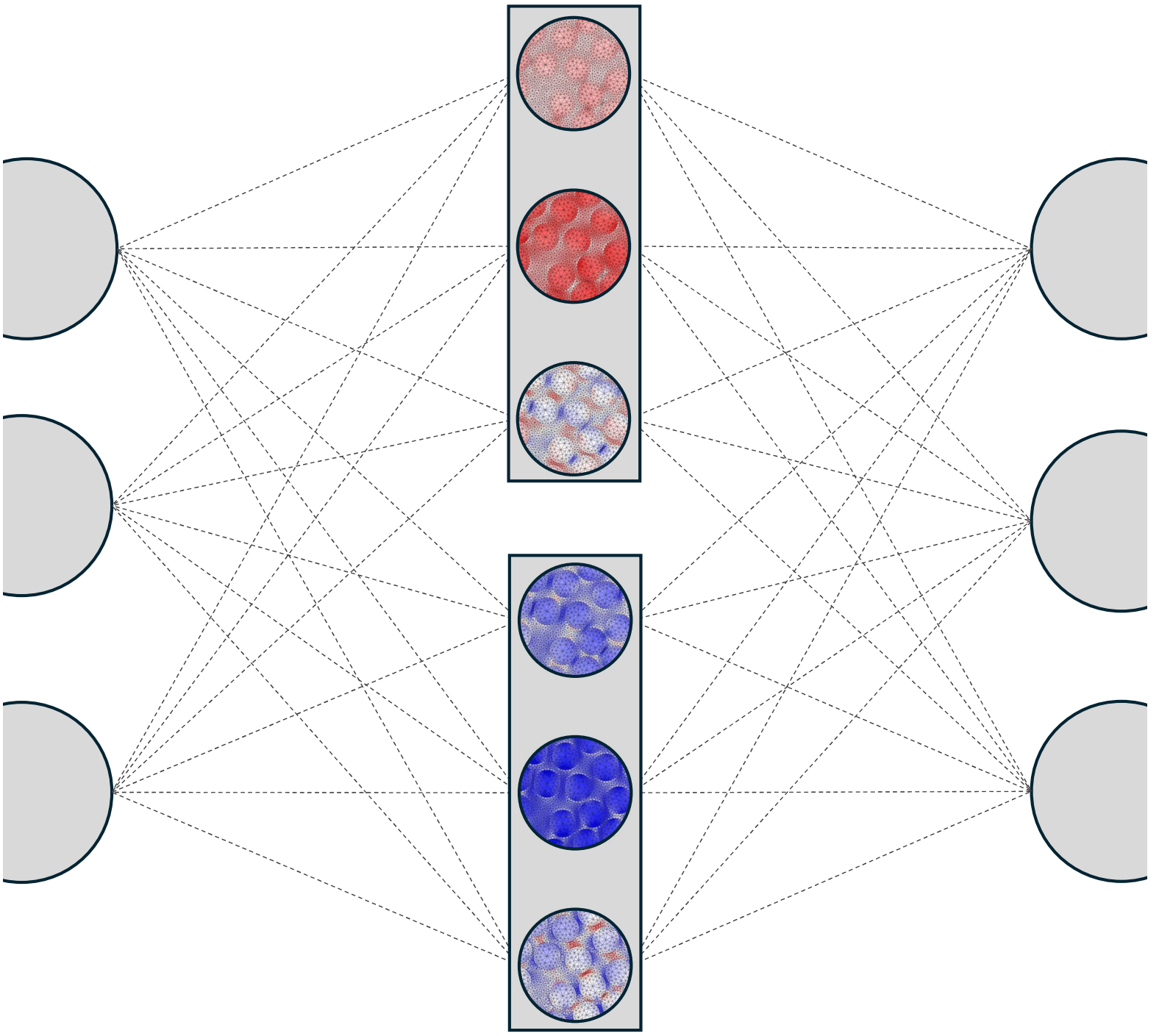
Predicting maximum stresses in composite microstructures using surrogate models



A.M.C.M. van Gils

TU Delft

Predicting maximum stresses in composite microstructures using surrogate models

MSc Thesis

by

A.M.C.M. van Gils

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 7, 2025 at 2:00 PM.

Student number: 4322924
Project duration: May 2024 - December 2024

Thesis committee:

Dr. I.B.C.M. Rocha	TU Delft, chair
M.A. Maia	TU Delft, daily supervisor
Dr. ir. F.P. van der Meer	TU Delft, supervisor
Dr.sc.nat M. Ramgraber	TU Delft, supervisor

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

TU Delft

Preface

One of many prerequisite courses to bridge the gap between a HBO Bachelor Civil Engineering (CE) and a Bachelor CE obtained at TU Delft was "*Computer programmeren HBO*". During this course the basics of Python programming was taught. Most of my fellow classmates despised the exercises as all they got was a screen filled with words that concluded with "**SyntaxError:** invalid syntax". As this class was in the midst of the COVID pandemic, students were stuck at home and it was difficult to receive help from tutors. ChatGPT or comparable Large Language Models (LLMs) were not publicly available, so one better hoped a relevant question was answered on stackoverflow (a forum for programmers). I, on the other hand, loved the challenge of finding the correct syntax and puzzling my way through the exercises. I have always been a big fan of puzzles; jigsaw puzzles, sudokus, you name it. During this class my passion for programming was born. Working with datasets, visualizing data or automating some simple tasks were all very beneficial exercises to learn about the endless possibilities with programming.

Most of the classes during my Master Civil Engineering at TU Delft involved programming as well. In group assignments I always made sure to be responsible for the programming work, even if this meant I was doing much more work than others. One of the final courses I took was "*Data Science and Artificial Intelligence for Engineers*" and this was the moment I knew I wanted to do research in this field. Iuri Rocha, the chair of my thesis, was involved in the course and after we talked a few times, we agreed on the interesting state-of-the-art research topic that forms my thesis.

That is why I would like to thank both Iuri Rocha and Frans van der Meer for the opportunity to perform this research with the SLIMM Lab. Your vision and knowledge were of tremendous help during the last few months. Furthermore I would like to thank my daily supervisor Marina Maia for keeping cool and always being patience and helpful. I wish you all the best with the final stretch of your PhD, I am sure you will be proud of the result.

A big thank you to all my friends, family and in-laws, who supported me for so many years. We came a long way, but in the end it was all worth it. Last but not least I would like to thank my girlfriend Quirine. We started our academic journey together and both experienced countless ups and downs. In the end, it came down to having a plan A and sticking to it. I'm so proud of your recent achievements and I hope to make you proud too!

Ruben van Gils

Den Haag, January 2025

Abstract

Composite materials are crucial for advancing sustainable engineering practices as they offer high strength-to-weight ratios, making them ideal for applications in aerospace, automotive and civil engineering. Meeting global sustainability goals demands a shift towards efficient materials, resulting in composites becoming increasingly significant. Besides difficulties in producing highly complex composites there are also challenges in computational methods to accurately predict the behavior of structures made from composite materials within a reasonable computational time frame.

In most civil engineering practices, the Finite Element Analysis (FEA) is a useful method to derive stress-strain paths. However, these methods often rely on homogenized material assumptions which fail to capture complex microstructural stresses in high-performance composites. A more suitable method to predict the behavior of these materials under complex loading conditions is available using multi-scale modeling techniques. This approach is essential but computationally expensive, especially when high accuracy at the microscale is needed. Surrogate models based on a machine learning framework have been investigated in prior research. These models typically make predictions on homogenized average stress at macroscale rather than the maximum stress at microscale, which identifies as the research gap. The maximum microscopic stress has been recently used as a failure indicator.

The main objective of this research is to design a surrogate model, based on the physically recurrent neural network (PRNN) designed in previous research, that can accurately predict the local maximum stress at the microscale in heterogeneous composite materials. Additionally, this research explores how knowledge from existing models that predict average stresses can be leveraged to enhance new models that predict local quantities at the microscale through transfer learning and multi-task learning techniques.

Among the three developed models, two models using an encoder-decoder architecture performed poorly because the decoder averages the results instead of being able to find the limits. The decoder was removed in the third model and this yielded promising results. This model maintains high fidelity in stress predictions with lower computational costs because the number of trainable parameters in this model is limited. Transfer learning yielded faster convergence but did overall not improve significantly in terms of error. The decrease in computational time is very limited when transfer learning is applied over direct training as the models used in this thesis are small and do not need extensive datasets or epochs to converge. The multi-task approach achieved very promising results as it enables accurate predictions for average and maximum stresses in a single model. Cross validation showed strong model robustness and flexibility across unseen loading scenarios.

Finally, a triple-task model showed that the PRNN is able to act as a modular framework as also the minimum microscopic stress can be accurately predicted without increasing the model and or training set size. The developed surrogate model shows that the architecture of the PRNN developed in previous research can be effectively modified to make predictions on other tasks. It provides a viable tool for simulating composite materials behavior at microscale.

Contents

List of Figures

1 Introduction

This chapter provides an introduction to this research. To start, Section 1.1 describes the background of this research topic. The main objective is discussed in Section 1.2 and the scope is presented in Section 1.3. The research questions are formulated in Section 1.4 and finally the structure of this thesis is elaborated in Section 1.5.

1.1. Context

The demand for composite materials is increasing in engineering applications. One of the reasons is because the world needs to adapt to using materials more efficiently in order to reach global sustainability goals set in 2050[1]. Resources on Earth are finite and the extraction of raw materials and the processing of these raw materials is often polluting and an irreversible process. On one hand all current materials that are deployed in systems and architectures need to be sustained. This is done with the principle of circularity where the value of a material in objects or systems is maintained throughout its life cycle[2]. On the other hand newly built objects and systems should be made from efficient materials.

An increasing amount of these materials are composites. A composite is a material that is produced from two or more different materials. Composites have the feature that it combines properties from multiple materials in one single material to obtain highly desirable properties. This technique of combining materials is something that was used as early as 3500 BC. By smelting tin and copper a much stronger and easier to cast material could be forged; bronze[3]. Popular composites of this day and age is carbon fiber, where carbon filament layers are used in combination with a binding polymer. This results in a very strong and lightweight material which is used in applications where the strength-to-weight ratio is an important aspect, like in aerospace engineering[4]. Composites are not often used in civil engineering due to high cost but this is likely change in the next few decades.

Besides difficulties in producing highly complex composites there are also challenges in computational methods to accurately predict the behavior of structures made from composite materials within a reasonable computational time frame. For structures that are not considered critical, composite material properties may be simplified. Analysis can be done with homogeneous material properties generating results that are sufficiently accurate[5]. However, for high performance structures like the superstructure of aerospace applications, ships and various applications in the civil engineering domain, higher-fidelity analyses are necessary. The mesomodel with homogeneous material properties is not able to predict the structural behavior accurately, as detailed information at microscale is lost. An analysis method that is used to account for microstructural details is multiscale modeling. This method analysis is done at multiple scales in time and/or space[6].

A suitable multiscale modeling technique for predicting the constitutive behavior of composite materials is FE^2 [7][8]. With FE^2 a structure at macroscale is discretized and at each integration point a microscopic model is linked. At both scales a Finite Element (FE) analysis is performed which explains the name for FE^2. As model complexity and structural scale increases, the computational cost increases such that FE^2 becomes intractable in engineering practice. By replacing the FE simulation at microscale by a surrogate model, computational time can be reduced, making multiscale modeling feasible for solving complex heterogeneous materials[9].

Finding a balance in computational time, accuracy and the amount of data that is needed to train these surrogate models, has become an active research field. The options for surrogate models are endless and often uniquely defined per task. In the field of constitutive modeling, examples of different architectures are the use of Recurrent Neural Networks (RNN) shown by Logarzo et al. [10] and Ghavamian and Simone [11]. With the use of RNN, path-dependency of heterogeneous materials can be accounted for by including additional parameters to learn how previous network's state(s) can be used to make a new

prediction. However, RNNs have the nature to be data-hungry, making it inapplicable in certain scenarios. Another class of neural networks is the Physics Informed Neural Network (PINN), which have been proposed by Raissi et al. [12] to solve partial differential equations. Additional physical relationships, such as initial and boundary conditions, are introduced in the network's loss function, resulting in the network converging faster to the target. PINNs are designed to be trained to satisfy both the given training data and the imposed governing equations. By doing so, a neural network can be guided without an extensive dataset.

Haghighat et al. [13] presented the application of PINNs to inversion and surrogate modeling in solid mechanics. In the case of inversion, structural or model parameters are often unknown and so measured or generated data is used to make predictions on said model parameters. Despite the success exhibited by the PINN approach, they found that it faces challenges when dealing with discontinuous solutions. The network architecture is less accurate on problems with localized high gradients as a result of discontinuities in the material properties or boundary conditions, like in the case of composite materials.

Another way to introduce physics into NNs has been explored by Maia et al. [14]. The architecture suggested in [14] is a neural network with embedded physics-based material models in a material layer. This network is trained on small datasets and is able to capture complex path-dependent behaviors through the history variables automatically handled by the material model, hence the name physically recurrent NN or PRNN. The approach outperforms state-of-the-art RNNs with significantly less data and speed-ups of four orders of magnitude in FE^2 problems. The surrogate model consists of an encoder to convert macroscopic strains at each integration point to fictitious strains. The material layer is responsible to convert local strains to local stresses using a material model, which takes into account the previous state of the material point through its own internal variables. The decoder converts the microscopic stress back to macroscopic stress which concludes a single iteration. To this moment, this architecture has been investigated only for predicting average stresses. Conceivably the embedded material models contain additional information on the microscopic state of the material.

1.2. Main research objective

A surrogate model that can accurately predict maximum values can give valuable insight in the behaviour of the heterogeneous material at microscale which translates to structural behaviour at macroscale. Specifically for the PRNN there are reasons to believe this works as this network is already able to accurately predict to complex unseen loading scenarios and captures path-dependency. This gives reasons for the main objective of this research to create a modified PRNN that is able to accurately predict local quantities at microscale such as the maximum stress.

A secondary objective is to see if transfer learning can be applied to use parameters from the original PRNN as input to the new model. This can speed up the learning and possibly lead to better predictions. Haghighat et al. [13] applied transfer learning to use a pre-trained PINN to perform training on new datasets with different parameters and concluded that training converged much faster with transfer learning. Cheung and Mirkhalaf [15] showed how transfer learning is used to generate a high-fidelity dataset by using a limited amount of high-fidelity data and a pre-trained RNN trained on low-fidelity data. In this work, transfer learning strategies will be investigated to leverage the learning from elements of the current PRNN, such as encoder, decoder and material points, into the new task of predicting maximum microscopic stresses.

Finally, a multi-task approach will be considered to see if multiple predictions can be made with a single model. Ban et al. [16] applied multi-task neural networks to predict multiple mechanical properties, like tensile strength and elongation, of steels. Iraki et al. [17] showed a multi-task learning-based optimization approach for finding diverse sets of material microstructures. In the end the proposed model should suffice with a limited training set, accurately predict unseen loading scenarios and be able to capture path-dependency.

1.3. Scope

This research focuses on composite materials consisting of elastic fibers with an elastoplastic matrix. The goal is to modify the PRNN by Maia et al. to predict the maximum hydrostatic stresses at microscale. Clarijs [18] used hydrostatic stresses as a criterion to predict embrittlement of glassy polymers and Wismans et al. [19] performed a micromechanical analysis of a short-fiber reinforced polymer of the local stress state. A 3D Representative Volume Elements (RVE) in combination with a constitutive model including hydrostatic stresses as a failure predictor is used that describes the intrinsic deformation response of the matrix under uniaxial and multiaxial loading conditions. This thesis however focuses on a single 2D RVE that is used to analyse the material behavior. Three datasets will be used during this thesis that follow the framework presented by Maia et al..

1.4. Research questions

In order to structure this research the following research question was formulated to address the main objective:

To what extent is a modified PRNN able to accurately predict local maximum stress values in heterogeneous microscopic material models?

To answer the main research question, several sub-questions are specified. A short comment is included to provide more context with each sub-question:

1. **How to efficiently incorporate the maximum stress in the network?** The newly generated data should be used with the current network to assess the accuracy of the current model. Do any modifications need be made on the current network to correctly process the maximum stress data? Does the meaning of the current encoder, material layer and decoder change?
2. **In what ways can transfer learning be applied to leverage the knowledge of existing models in creating new models?**
   Various studies [20] [21] [22] have shown that transfer learning can be an efficient technique to use a trained model to perform other tasks.
3. **Can the architecture be leveraged to a multi-task approach?**
   The main focus of this research is to design a model that predicts maximum stresses. In case this model performs well, there are now two separate models needed to predict average and maximum stresses. Can these tasks be combined in a single multi-task model?

1.5. Thesis structure

This thesis is structured as follows:

- The theoretical background is elaborated in Chapter 2.
- Data generation is presented in Chapter 3.
- Designing a model that predicts the local maximum stress is showcased in Chapter 4.
- Chapter 5 presents the transfer learning strategies used to go from one task to another (e.g. from average stress prediction to maximum microscopic hydrostatic stress).
- Combining tasks in a multi-task approach is investigated in Chapter 6 and Chapter 7.

Following these results, conclusions and recommendations will be presented.

2 | Theoretical framework

The theoretical background that is needed for this thesis is presented in this chapter. The basis forms the Finite Elements Method and the FE^2. Machine learning techniques and methods are presented as well and these topics are the backbone of this research.

2.1. FEM

The Finite Element Method (FEM) is a popular method to perform structural analysis. The displacement and it's derivatives are the unknowns. First, the structure is discretized in space to create a mesh, which is the numerical domain for the solution. The mesh consists of elements and nodes. Each element in the mesh represents a small finite part of the structure and gives rise to additional equations. Integration points are defined that live within the elements. Stresses are calculated at the integration points and extrapolated to the nodes. The stiffness and other properties related to the material are obtained from the definition of the constitutive model, which relates strains to stresses. These properties must be known a priori to be able to solve the governing equation:

$$\mathbf{Ku} = \mathbf{f} \tag{2.1}$$

Here **K** represents the global stiffness matrix, **u** is the unknown nodal displacement vector and **f** is the load vector. When this system of equations is solved, all displacements at the nodes are known and the displacement and stresses at any other point of the structure can be found by interpolation between nodes. By combining the equations of all elements with Dirichlet and Neumann boundary conditions, a system of differential equation arises. Dirichlet boundary conditions specify the solution along a boundary of the domain whereas Neumann boundary conditions specify the derivative of the solution. The solution at each integration point is approximated by solving the system of equations numerically. This method works well for homogeneous materials as a single constitutive model is typically assigned to the entire domain. Composite materials however do not have this characteristic of equal material properties over its domain. Discontinuities can exist at the microscopic fiber-matrix interface and both the distribution and direction of fibers vary over the domain. FEM can still be used to analyse the general structural behavior, but a more complex model is needed to capture more intricate material behavior that is otherwise lost when homogenizing material properties.

2.2. FE^2

Section 2.1 concluded that FEM is a suitable method to analyse materials but the problem with composites is the separation of scales in which the length scale of the microstructure is too small to model entirely. The multiscale finite element analysis (FE^2) method is a popular method to model composites. The name suggests that the analysis is done at multiple scales: at structural macroscale (Ω) and at microscale (ω) where fiber-matrix structures are modeled. The microscale models are implemented at each integration point of the grid at macroscale as this is where the solution is approximated. The macroscopic strains ε^{Ω} act as boundary conditions for each microscale model. The FE analysis at microscale outputs a field of microscopic strains ε^{ω} and microscopic stresses σ^{ω}. After homogenizing the microscopic stresses, a single stress value is sent back to the macroscale model and the global stresses σ^{Ω} are computed. Finally, global equilibrium can be solved. The comparison of FEM and FE^2 is shown in the schematic overview in Figure 2.1.

For each iteration of the FE analysis at macroscale, a finite number of calculations need to be made for each model at microscale, one at each integration point. This nested number of calculations causes the computational time to be amplified greatly, often leading to prohibitive computational costs.

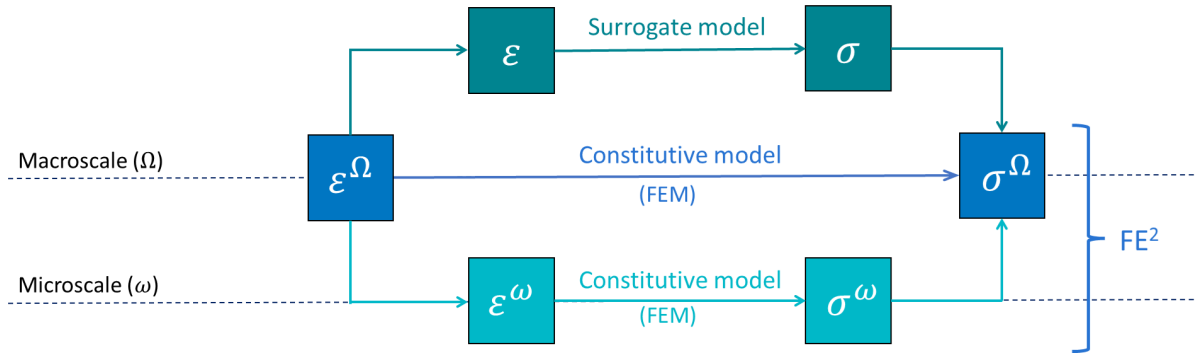


Figure 2.1: FEM, FE^2 and surrogate models at different scales

2.3. Machine Learning

One way to improve FE^2 is by substituting the computationally expensive microscale models with a mathematical model that is trained through machine learning, indicated by the green block in Figure 2.1. Machine learning is a subset of Artificial Intelligence (AI) that uses statistical learning algorithms to recognise patterns in data. A human being is often able to see patterns in 2D or 3D, but once the data is highly dimensional, this becomes very challenging. As a computer processes data completely differently than a human does, it is able to learn patterns in higher dimensions. As promising as this sounds, there are drawbacks like the *curse of dimensionality*. This refers to difficulties like the exponential increase in computational time, data sparsity and model overfitting that arise when data becomes highly dimensional. Data sparsity is when the data points are widely distributed in a high-dimensional space, making patterns harder to observe. Regularization techniques help to prevent overfitting and enforce sparsity in the model parameters. The availability of *enough* useful and diverse data is often a challenge. The goal of this research is mostly to reduce computational time and thus to limit the amount of data needed.

This section presents Neural Networks, a popular machine learning technique that will be applied in this research. First, a basic network is presented and then more complex variations will be investigated.

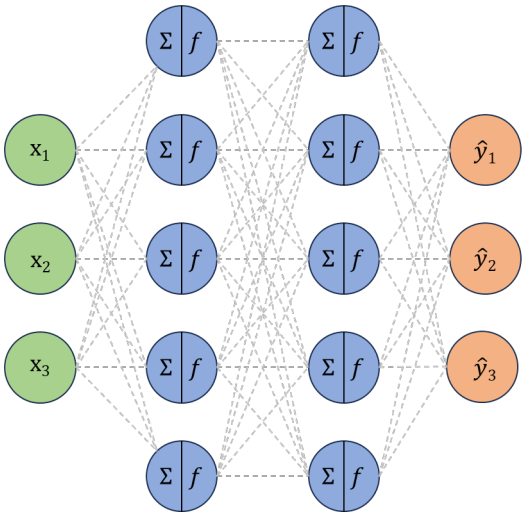


Figure 2.2: A visual representation of a Neural Network

Neural Networks

As the name suggests, a Neural Network (NN) is a network created by connecting layers of neurons. Conceptually, there are similarities to the brain as information is transferred through interconnected neurons. Figure 2.2 shows a visual representation of a Neural Network with an input layer (3 neurons: x_1, x_2, x_3), 2 hidden layers (5 neurons each) and an output layer (3 neurons: $\hat{y}_1$, $\hat{y}_2$, $\hat{y}_3$). The neurons of each layer are connected with all neurons of the next layer, indicated by the grey dashed lines. Data flows from the input layers, through the hidden layers to the output layer and at mathematical operations are done at each layer. With respect to this thesis, the macroscopic strains functions as input data and x_1, x_2, x_3 can be rewritten as $\varepsilon^{\Omega}_{xx}$, $\varepsilon^{\Omega}_{yy}$, $\varepsilon^{\Omega}_{xy}$.

The hidden layers are responsible to learn intricate patterns in data which is the core strength of this network. Two operations are applied at the neurons in the hidden layers, as indicated in Figure 2.2. First, Σ represents the input which is a linear combination of all neurons in the previous layer ($\mathbf{a_{l-1}}$), multiplied by a weight ($\mathbf{W_l}$) and an added bias term ($\mathbf{b_l}$).

$$\mathbf{v_l} = \mathbf{W_l a_{l-1}} + \mathbf{b_l} \tag{2.2}$$

The weights are scalars that indicate the importance of a certain neuron in the previous layer. The bias term is added to be able to shift the function $\mathbf{v_l}$ to be able to better fit the data. The effect of the weights and bias is shown in Figure 2.3.

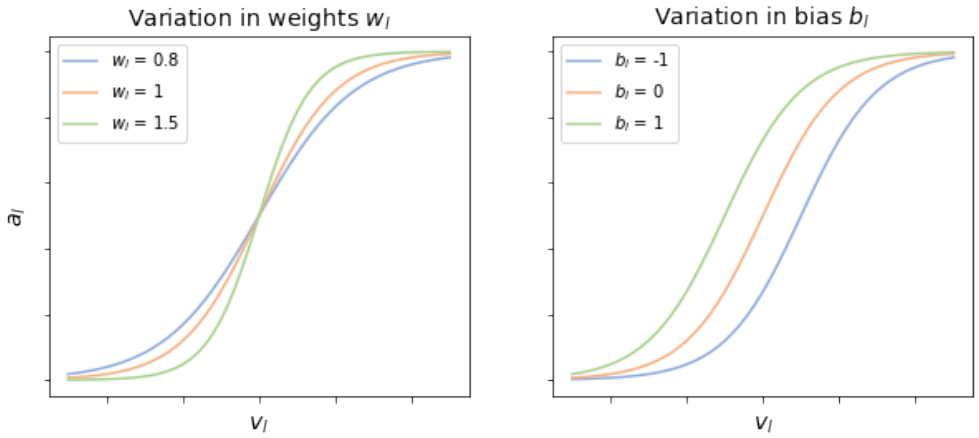


Figure 2.3: Weight w_l and bias b_l in a Neural Networks hidden layer

The second operation is to apply an activation function f over $\mathbf{v_l}$ that was obtained in equation 2.2 at each neuron. An example of an activation function is the ReLU activation function, which only outputs positive numbers. ReLU outputs 0 in case $\mathbf{v_l} \leq 0$ and it gives the value of $\mathbf{v_l}$ in case $\mathbf{v_l} > 0$. The activation function is a way to introduce non-linearity to the system. The output of the neuron is $\mathbf{a_l}$ and is given by:

$$\mathbf{a_l} = f(\mathbf{v_l}) \tag{2.3}$$

The final layer, in Figure 2.2 indicated by $\hat{y}_1$, $\hat{y}_2$ and $\hat{y}_3$, is the output layer. For this thesis, the task of the network is to predict the maximum microscopic stresses and so the final layer can be modeled by 3 neurons: $\hat{\sigma}^{\Omega}_{xx}$, $\hat{\sigma}^{\Omega}_{yy}$, $\hat{\sigma}^{\Omega}_{xy}$.

Besides parameters like the number of hidden layers and the amount of neurons in each layer, the predictions of the strains by the network is dependent on the weights ($\mathbf{W_l}$) and biases ($\mathbf{b_l}$). These parameters are assigned a random value before the network is trained. The training of the network happens when data is seen by the network and the predictions are compared to the known true value,

also known as the target $\mathbf{y}_i$. A popular loss function is the mean squared error (MSE) which is introduced in equation 2.4. The MSE measures the error by assessing the average squared difference between the target value $\mathbf{y}_i$ and the model prediction $\hat{\mathbf{y}}(\mathbf{x}_i)$ over N samples. In case there is no error, the MSE equals zero. This suggests that the value of the MSE, or any other loss function, should be minimized in order to obtain a model with high accuracy.

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(\mathbf{y}_i - \hat{\mathbf{y}}(\mathbf{x}_i))^2 \tag{2.4}$$

A risk that has to be taken into account when optimizing the model parameters is overfitting. Overfitting occurs when the network is trying to fit the data, or noise in data, too well and is unable to generalize to new or unseen data. Techniques exist to prevent from overfitting. Often, the dataset is split up in a training set and a validation set. The model parameters are trained with the training dataset and then the best model is validated by an unseen validation set. The validation loss is the error between the predicted output of the model and the target value on the validation dataset which is unseen during training. The model that has the lowest validation loss, also generalization loss, is often considered optimal, as the chance of under- or overfitting are lowest while remaining accurate in its predictions. This balance can be seen in Figure 2.4.

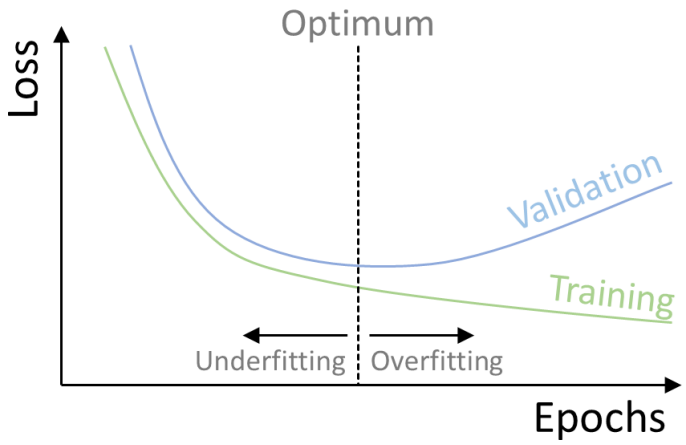


Figure 2.4: Training loss and validation loss

The process described above is feeding data through a network and so the name feed-forward Neural Network was formed. Until now, the process of updating the weights has not been touched yet. This process causes the actual output to be closer to the target and is often done by an optimization algorithm called Stochastic Gradient Descent (SGD). In short, the algorithm uses gradient of the loss function to then update the weights in the direction that causes the function to minimize. A popular method to derive the gradient with respect to variables is through backpropagation. The change in weights, among other variables, is controlled by a parameter called the learning rate. This variable decides the amplitude of the change in weights. A large value of learning rate can decrease the amount of training that is needed, but if the value is too high, the network might not converge to the global minimum of the loss function. On the other hand, a small value of learning rate will cause training to cost a lot computational wise, which is not desirable.

A feed-forward Neural Network is the basis of more complex NNs. For a lot of tasks this network suffices, but when the complexity of the data increases, this network often needs too much data or does not converge to the target. For the task in this thesis, the complexity lies in the non-linear load path, path-dependency and the non-homogeneous material model. This path-dependent behaviour is an important aspect that can be captured with Recurrent Neural Networks.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of Neural Network to process sequential data such as a time series. The structural response due to a non-linear load is an example of time series. RNNs have the ability to store past information in order to make predictions about future data points. The storage of historic data is captured in a hidden state which acts as a memory. This memory is indicated with a black box in the left visual representation of a RNN in Figure 2.5. The representation on the right in Figure 2.5 shows the expanded version which showcases the dependency on the previous state more clearly. This hidden state allows the network to capture temporal dependencies which are relevant in this thesis. The equations have a similar setup as equations 2.2 and 2.3 in feed-forward NNs, but the terms related to the hidden state are added as can be seen in equations 2.5 and 2.6.

$$\mathbf{h}^t = \phi(\mathbf{W}_1\mathbf{x}^t + \mathbf{W}_s\mathbf{h}^{t-1} + \mathbf{b}_s) \tag{2.5}$$

$$\hat{\mathbf{y}}^t = \phi(\mathbf{W}_2\mathbf{h}^t + \mathbf{b}_2) \tag{2.6}$$

Another feature of RNNs is that the weights between different time steps are shared. This greatly reduces the amount of parameters the network needs to learn which is advantageous. Another advantage is that RNNs can process time series of any length. This is helpful as the length of a time series may vary, or is simply unknown a priori.

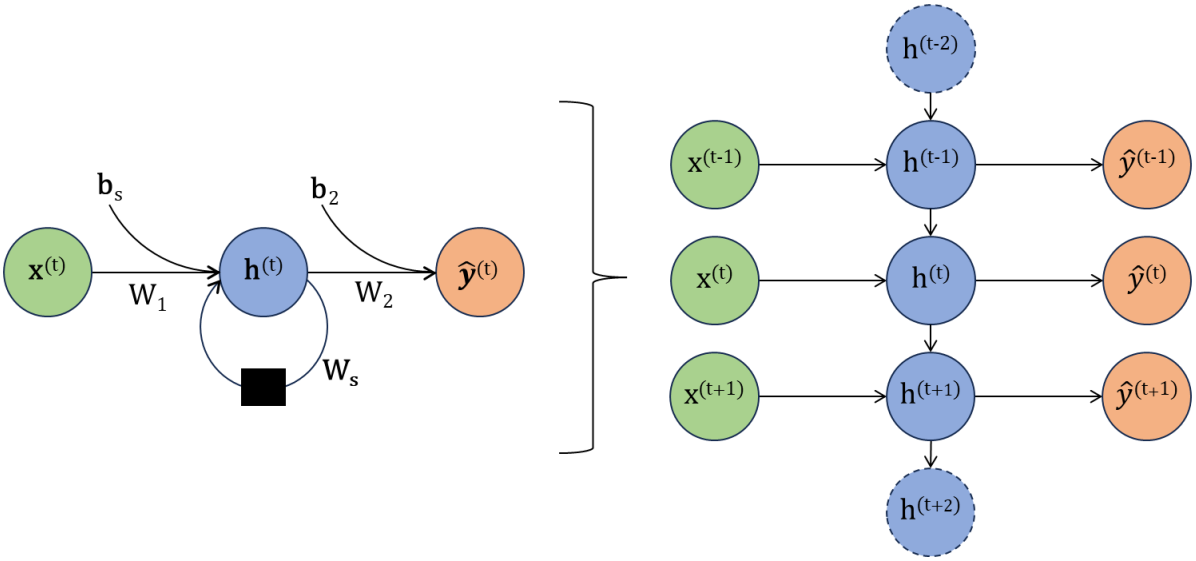


Figure 2.5: Recurrent Neural Network - two schematic representations

Besides the potential of RNNs, there are things to be wary of. As with feed-forward NNs, a RNN uses an algorithm, like backpropagation, to find the gradients with respect to the target function in order to update its weights. As the RNN uses all previous timesteps through the hidden state, the contributions of each time step to the gradient is multiplied through the chain rule. This causes the gradient to either shrink to zero (*vanishing gradient*), or explode to infinity (*exploding gradient*).

A solution to this problem is to control the memory by the introduction of special cells. These cells consist of gates that can either pass or forget information. This selective retention and omission of data is crucial in maintaining relevant information, especially over long sequences. Gated Recurrent Unit (GRU) and the Long-Short Term Memory (LSTM) are two popular methods that apply these cells. The gates have trainable parameters that need to be optimized, just like the weights for the NN itself.

The amount of historic data the cells needs to let through is dependent on the complexity of the data. Often the amount of variables associated with the data is highly dimensional and the effect of any variable can be dependent on many things like amplitude and frequency. An example is when one wants to predict the air temperature. Frequency dependent variations like day and night and summer and winter

can be thought of, but there are much more complex natural phenomena that effect the air temperature like ocean currents and the amount of greenhouse gases in the atmosphere.

Due to the complexity of the data in this thesis and the architecture of RNNs being prone to overfitting, there are better architectures to be considered. A Physically Recurrent Neural Network can lead to promising results with less needed data which is discussed in section 2.3.

Physically Recurrent Neural Network

In this section, the hybrid architecture proposed in Maia et al. is briefly summarized. The main components of PRNNs includes a data-driven encoder, a material layer with embedded physics-based material models and a decoder.

Encoder

The input for this architecture consists of the macroscopic strains $\vec{\varepsilon}^{\Omega}$ at the integration points of the macrostructure. The input is sent through an encoder, which consists of an arbitrary number of (dense) layers of neurons with conventional activation functions. The encoder outputs data that live in latent space, which is an abstract higher-dimensional representation of several microscopic integration points each, with dimension 3 (for 2D problems). This way, the output of the encoder can be interpreted as the strains of fictitious material points. In this work, the encoder consists of a single dense layer.

Material layer

The material layer is the main identity of this architecture. The output of the encoder is forwarded to the material layer, in which the material models are introduced. A material model is a set of *m* neurons, where *m* equals the input size. In case of 2D modelling, this results in three stress components and so the material models are a group of 3 neurons, which represent the three stresses. Each of these groups represents a fictitious integration point in which a constitutive model $\mathcal{D}^{\omega}$ is nested that convert the fictitious strains (ε) to stresses (σ). The implementation of the material layer is based on the idea that the output of the encoder is an actual representation of strains, instead of some unknown variable in latent space. This (partly) resolves the idea of the network as being a black-box.

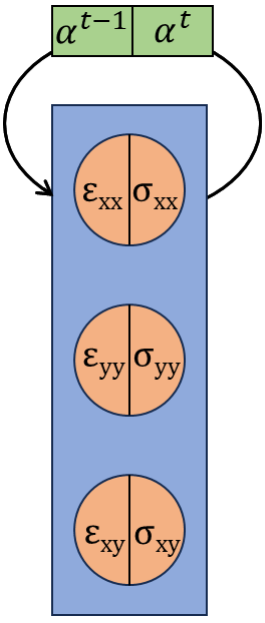


Figure 2.6: PRNN material model

Besides converting strains to stresses, the material model also stores its current state in an internal variable $\vec{\alpha}^t$ at timestep t, as can be seen in Figure 2.6. These variables are then used as input, along with the fictitious strain, in the following time step, allowing path-dependency to arise naturally. This operation is shown in equation 2.7.

$$\vec{\sigma}, \vec{\alpha}^t = \mathcal{D}^\omega(\vec{\varepsilon}, \vec{\alpha}^{t-1}) \tag{2.7}$$

Decoder
The material layer then forwards the data to the decoder. The decoder does the opposite of the encoder as it converts data from latent space back to its original dimensionality and complexity. In this case the decoder converts the stresses at the fictitious material points to the predicted stresses at the integration points of the macrostructure $\vec{\sigma}^\Omega$. This process is an analogy to the homogenization operator that transforms the microscale stresses to macroscale stresses. The decoder in [14] consists of a dense layer with a SoftPlus activation function applied on the weights. This is done to represent the homogenization process through numerical integration in which weights are strictly positive. The output of the decoder is then used as output of the network.

2.4. Transfer Learning

In the previous section a couple of different machine learning architectures have been shown. It is mentioned that these models often need huge training datasets, hindering their training procedure. It is a tedious task to create new model from scratch when already existing and proven models have near similar tasks. Transfer learning is a method where an existing model is used as input for a new model or a new task. This can drastically improve training a model for new tasks, especially when the previous model was trained on an extensive dataset. The training of the new dataset is likely to start with a better accuracy, converges quicker and might be able to reach a higher performance than training without transfer learning.

3 | Data

In the domain of Computer Science, the phrase "*garbage in, garbage out*" is used to describe the importance of having high-quality, relevant and plentiful data. Data is the input to machine learning models and so flawed, biased or poor quality data will result in poor models. Data enables the learning process of a model, allowing models to identify patterns, make predictions and to improve over time. For any model to generalize well and to perform robustly, the quantity and diversity of training data is to be considered. This chapter describes the process of the data generation and what type of data is generated.

3.1. Micromodel

The complexity of a composite material at macroscale can be captured at microscale by a Representative Volume Element (RVE). A RVE is defined as the minimum volume of a sample required from which the material properties are independent of the size of the sample[23]. This suggests that the behavior of the RVE is to be the same as a macroscale model with equal initial and boundary conditions. This is very useful as a single micromodel suffices to generate all randomized stress-strain paths.

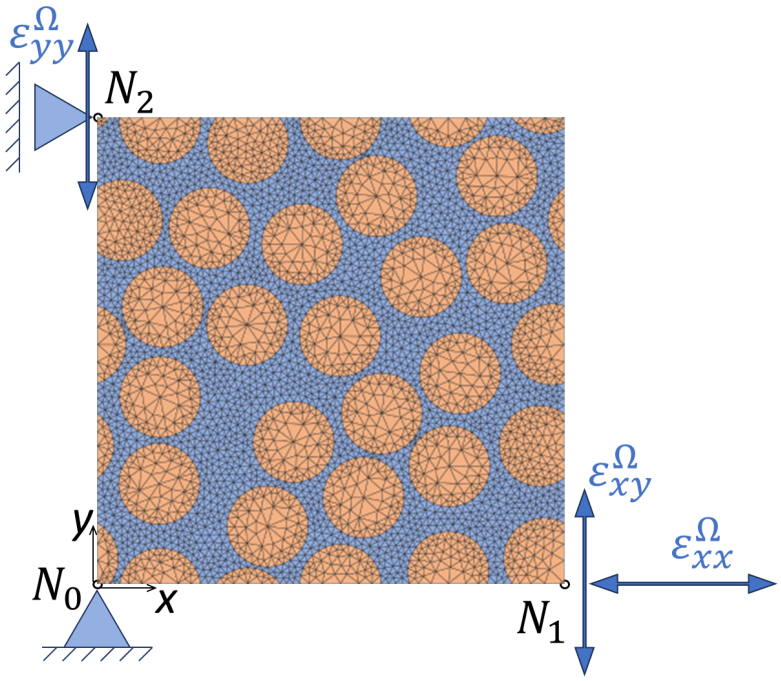


Figure 3.1: 2D Representative Volume Element adopted in this work

The RVE in question is shown in Figure 3.1. This micromodel contains 36 fibers (shown in orange) that are encapsulated in a matrix (shown in blue). The fibers have the following elastic material properties: E = 74000 MPa and $\nu = 0.2$. Elasticity in material science is the ability of a deformed material under loading to revert to its initial state once loading is removed. The matrix is modeled with elastoplastic properties with isotropic hardening. Elastoplasticity refers to materials that, dependently on the magnitude of the loading, have both elastic as plastic deformation. Plasticity is when a material under loading does not deform to its initial state when loading is removed. The yield point is the critical stress value at which a material transitions from elastic to plastic behavior. Isotropic hardening refers to the uniform expansion (in all directions) of the yield surface as the material undergoes additional plastic strain.

The elastoplastic matrix is modeled with the von Mises yield criterion, which states that plastic yielding

begins when the J_2 stress deviator invariant reaches a critical value[24]. In other words, this criterion is able to describe the yielding even during complex loading situations where a combination of compression, tension and shear are applied. The material properties are E = 3130 MPa, ν = 0.3 and the yield stress function is given by: $\sigma_y = 64.8 - 33.6exp^{-\epsilon^p_{eq}/0.0003407}$, where ϵ^p_{eq} is the equivalent plastic strain which is a scalar quantity that represents the accumulated irreversible plastic deformation[24], with ϵ^p_{eq} given by:

$$\epsilon^p_{eq} = \sqrt{\frac{2}{3}\epsilon^p_{ij}\epsilon^p_{ij}} \tag{3.1}$$

Furthermore, plane stress is assumed for both the fibers and the matrix as out of plane stresses are assumed to be negligible.

The micromodel is discretized in space by applying a mesh. The choice of triangular elements is evident for most 2D cases. A triangular shape is versatile and can represent complex curved shapes, which is relevant for the circular fiber components. The model consists of 3638 nodes and 7088 elements, which is split up in 3745 matrix elements and 3343 fiber elements.

Finally, boundary and initial conditions (BC, IC) are applied. The ICs are such that the structure is at rest initially. Dirichlet BCs in this model are applied at node 0 and 1. Displacements in both x- and y directions are limited at node 0. Node 2 has a limitation of displacements in x-direction. Neumann BCs are applied at node 1 and node 2. Node 1 can be displaced in both x- and y direction. A displacement in y-direction causes shear and a displacement in x-direction causes, dependently of direction, compression or tension. The BC at Node 2 is in y-direction and gives either compression or tension as well. The BCs are indicated in Figure 3.1.

3.2. Loading

The magnitude and direction of the loading over time are defined in this section and are presented by the variables $\varepsilon^{\Omega}_{xx}$, $\varepsilon^{\Omega}_{yy}$, $\varepsilon^{\Omega}_{xy}$. The articles that have lead up to this research started with a *canonical dataset*[25] or *Type I* loading[14] of 18 proportional loading directions. These include directions with uniaxial strains, pure shear, biaxial cases and biaxial with shear cases. Proportional suggests that a constant strain value ($\Delta s = 1.667x10^{-3}$) is applied after each timestep. In both works, 60 timesteps were used for each loading path. This dataset was a good starting point as the stress-strain results are known for canonical problems.

Complex composite structures often experiences complex loading scenarios. The *canonical* and *Type I* dataset is not diverse enough to be able to capture said complex loading scenarios. A load can be applied from any direction so a dataset with strain paths with proportional random directions is to be created. This is also referred to as *Type II* loading by Maia et al.[14]. In this work, 1000 samples are generated with proportional random directions. Both *Type I* and *Type II* are referred to as *monotonic* loading. The loading function is displayed in the first plot of Figure 3.2.

The second dataset that is used in this work is for non-monotonic loading paths which consists of a linear piecewise function. The strain paths in this dataset have a fixed moment in time (at timestep t = 40 [-]) when unloading takes place. At t = 50 [-], the loading direction is reversed again. This dataset is predominantly used as a validation set, so only 500 stress-strain paths were created. The stepsize and the number of timesteps is the same as for the monotonic datasets. The loading function is displayed in the second plot of Figure 3.2.

Finally, a more general approach was taken to create more diverse non-proportional loading paths. This approach, also known as random walks, samples random strain increments with random loading direction for each timestep. Each strain component is sampled from Gaussian Processes with $X \sim \mathcal{N}(\mu, \sigma^2)$ and a covariance function given by:

$$k(x_p, x_q) = \sigma_f^2 exp\Big(-\frac{1}{2l^2}\|x_p - x_q\|^2\Big) \tag{3.2}$$

where x_p and x_q are the time step indices of the sequence of loading function values, the variance σ_f^2 determines the range of the step size and the smoothness of the strainpath is described by the lengthscale l. The work of Maia et al. showcases the full algorithm that was used to generate random loading paths with GPs. Over 2500 strain-strain paths of this type were generated in this work. The loading function is displayed in the final plot of Figure 3.2.

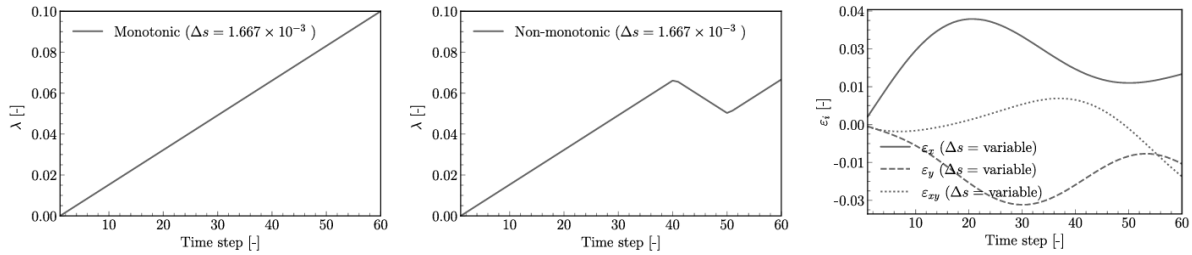


Figure 3.2: Proportional and non-proportional loading functions considered to generate the datasets

3.3. Generating data

The full-field stresses can now be generated by using FEM with the RVE and a sampled strain path as input. This research focuses on the stresses in the matrix, resulting in 3745 stress values at each timestep. As the maximum stress is a single quantity, the stress components (σ_{xx}^{ω}, σ_{yy}^{ω}, σ_{xy}^{ω}) are used to compute the hydrostatic stress:

$$\sigma_{\text{hydro}}^{\omega} = \frac{\sigma_{xx}^{\omega} + \sigma_{yy}^{\omega}}{2} \tag{3.3}$$

This represents the isotropic part of the stress tensor, associated with compression (or tension) without distortion. Initially the von Mises stresses were computed, but this resulted in a dataset that lacked in diversity as the data was purely positive and the maximum stress over time was mostly constant. The hydrostatic stress data is more diverse, which is beneficial for machine learning operations.

For each path, the maximum stress at each timestep is found by looping through time and applying a function on the data to filter the maximum value. Figure 3.3 shows a selection of 500 samples of each dataset, from which 5 samples are colored to showcase their unique features more clearly.

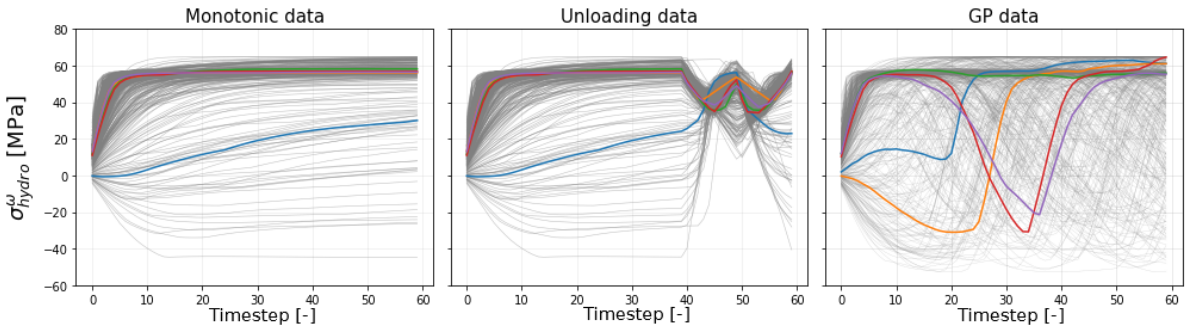


Figure 3.3: Maximum stresses $\sigma_{\text{hydro}}^{\omega}$ for different datasets of 500 samples

The full-field stress evolution of a random GP sample is shown in Figure 3.4. A snapshot at 8 different timesteps show how the stresses develop over time. The colorbar indicates the tensile and compressive stresses that occur in the sample. Figure 3.5 indicates the range of stresses that occur at each timestep more clearly with the same color gradient for the fiber and the matrix that was used in Figure 3.4. The timesteps selected in Figure 3.4 match the positions of the vertical lines in Figures 3.5a and

3.5b to be able to compare the stress ranges. On top of the vertical dashed lines, the stress distribution is plotted. This distribution indicates the probability density of a specific stress value at a timestep. The maximum stress in the matrix is the main topic of research and so the upper curve in Figure 3.5b will be used as input to the machine learning model that are discussed in later chapters.

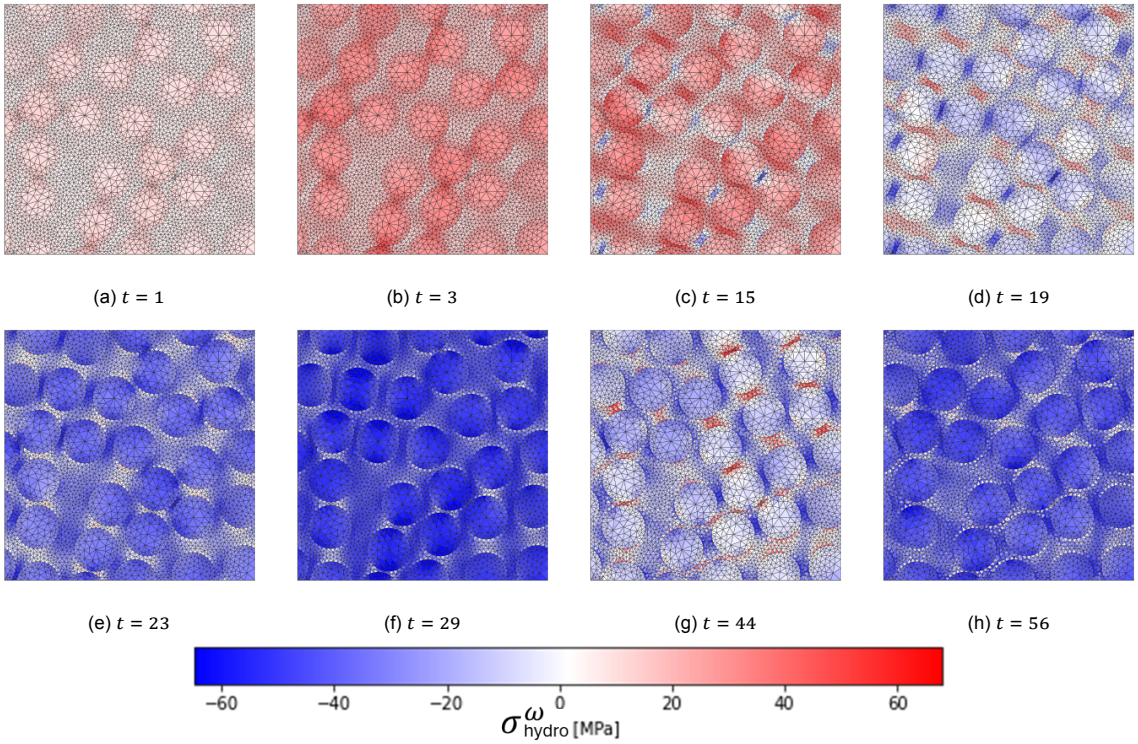
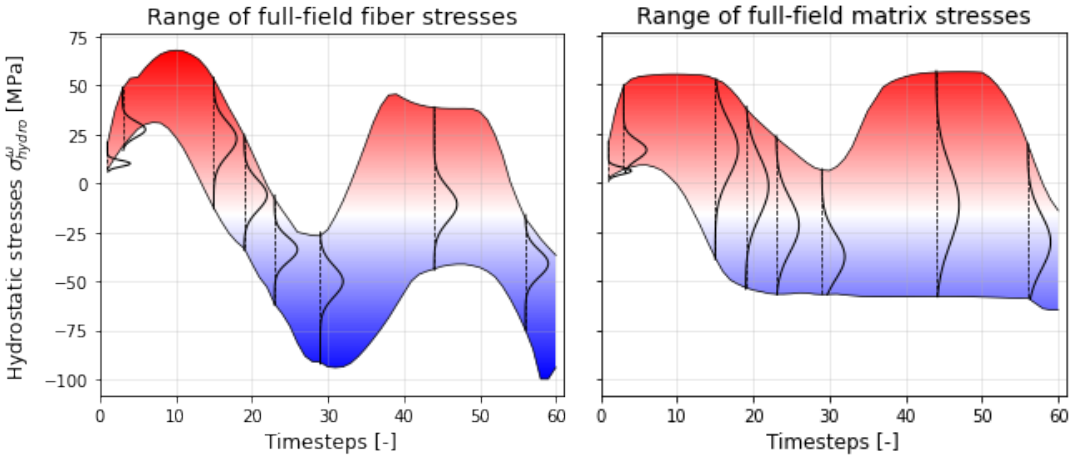


Figure 3.4: The full-field hydrostatic stress $\sigma^{\omega}_{\text{hydro}}$ evolution of a random GP sample



(a) Range of fiber stresses at each timestep

(b) Range of matrix stresses at each timestep

Figure 3.5: The hydrostatic stress $\sigma^{\omega}_{\text{hydro}}$ ranges of a random GP sample over time including the range of stresses at each snapshot of Figure 3.4 indicated by the vertical dashed lines

One observation that can be made on the datasets is that the concentration of hydrostatic stress $\sigma^{\omega}_{\text{hydro}}$ in the matrix is around 65 MPa for all datasets. The difference in material properties between the matrix and the fibers causes the stress in the matrix to reach its capacity while the fibers can still take up more

load. This observation is made visual by Figure 3.6, where a random sample from the monotonic dataset was taken. The blue and red lines refer to the stresses in the matrix and fibers respectively. The continuous blue line stagnates around timestep t = 10 [-] while the red line is still increasing. This can also be seen in Figure 3.5, where the range of full-field matrix stresses show a clear limit.

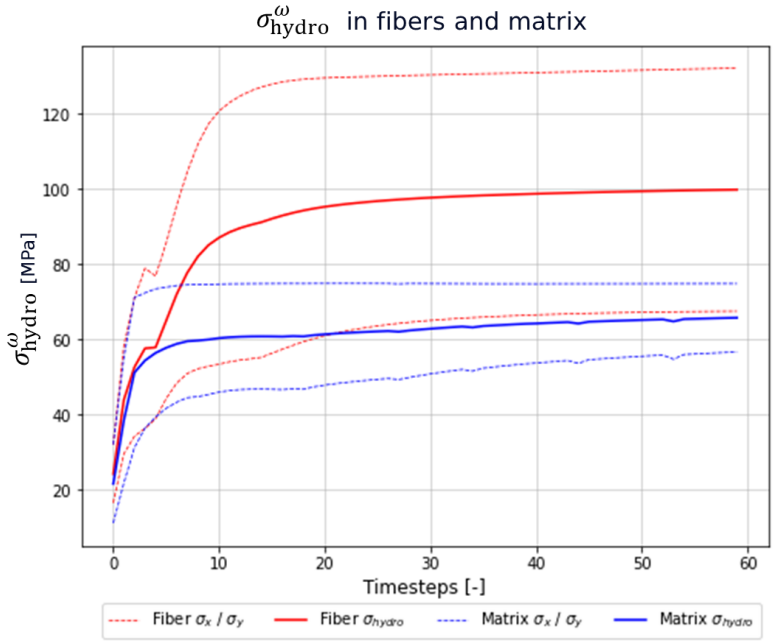


Figure 3.6: The difference in maximum hydrostatic stress $\sigma^{\omega}_{\text{hydro}}$ in the fibers and matrix

Furthermore, all datasets show a smooth curve. Both monotonic and non-monotonic datasets show a smooth curve as a constant strain increment is applied at each timestep. The GP dataset is created with lengthscale parameter l as an input, which is the parameter influencing the smoothness of the curve. Smooth curves often benefit training of machine learning models because the stress evolution is more predictable. However, the network should also be able to make accurate predictions in case the strain increments drastically. A prime example of a sudden change is an impulse, but this is not investigated in this research.

Table 3.1: Representation of how the data is structured

		$\varepsilon^{\Omega}_{xx}$	$\varepsilon^{\Omega}_{yy}$	$\varepsilon^{\Omega}_{xy}$	σ^{Ω}_{xx}	σ^{Ω}_{yy}	σ^{Ω}_{xy}	$\sigma^{\omega,\max}_{xx}$	$\sigma^{\omega,\max}_{yy}$	$\sigma^{\omega,\max}_{xy}$	$\sigma^{\omega,\max}_{\text{hydro}}$
Sample 1	Timestep 1										
	...										
	Timestep 60										
Sample N	Timestep 1										
	...										
	Timestep 60										

Table 3.1 shows the template in which the data is structured. The first 3 columns of the data show the input of the model which is the strain values ($\varepsilon^{\Omega}_{xx}$, $\varepsilon^{\Omega}_{yy}$ and $\varepsilon^{\Omega}_{xy}$). The work of Maia et al. is based on averaged stresses (σ^{Ω}_{xx}, σ^{Ω}_{yy}, σ^{Ω}_{xy}) and as transfer learning is one aspect of this work, the average stresses are needed. The averaged stresses are appended in columns 4-6. The maximum hydrostatic

stress ($\sigma_{\text{hydro}}^{\omega,\max}$) is based on the full-field stresses components at each level. The maximum hydrostatic stress and its components ($\sigma_{xx}^{\omega,\max}$, $\sigma_{yy}^{\omega,\max}$, $\sigma_{xy}^{\omega,\max}$) are appended to the matrix in columns 7-10. All stress values are in MPa and the strain values are unitless. The column and row headers are indicative, as only the values are used to form an array with floats.

4 | Designing the PRNN

In this chapter, 3 variants of the original PRNN are investigated for predicting the maximum hydrostatic microscopic stress. Further chapters will then be built on top of the best performing architecture that is found in this chapter.

4.1. Model A: Computing hydrostatic stresses after decoding

The prediction of the original PRNN is a [3x1] tensor that consists of the predicted average stresses $[\hat{\sigma}^{\Omega}_{xx}, \hat{\sigma}^{\Omega}_{yy}, \hat{\sigma}^{\Omega}_{xy}]$. As the maximum hydrostatic stress is a tensor with size [1x1], the output of the PRNN has to be modified. This first variant will compute the hydrostatic stress $\hat{\sigma}^{\omega,\max}_{\text{hydro}}$ before the loss function is computed. The target of the loss function is the $\sigma^{\omega,\max}_{\text{hydro}}$ that is presented in the final column of Table 3.1. Figure 4.1 shows the architecture of this proposed network. The green nodes are the three macro strain components that are the input. The material layer embeds the J_2 material models that are shown in the blue nodes. These are the same as the original PRNN. The layer with the orange nodes is the output of the network, which in the original model consisted of macroscopic stresses, while here it consists of the microscopic stresses that lead to the maximum hydrostatic stress. All dashed lines indicate the connection between nodes where a weight and activation function is applied, while the solid lines connecting the predicted stresses with the final node means that equation 3.3 is applied directly.

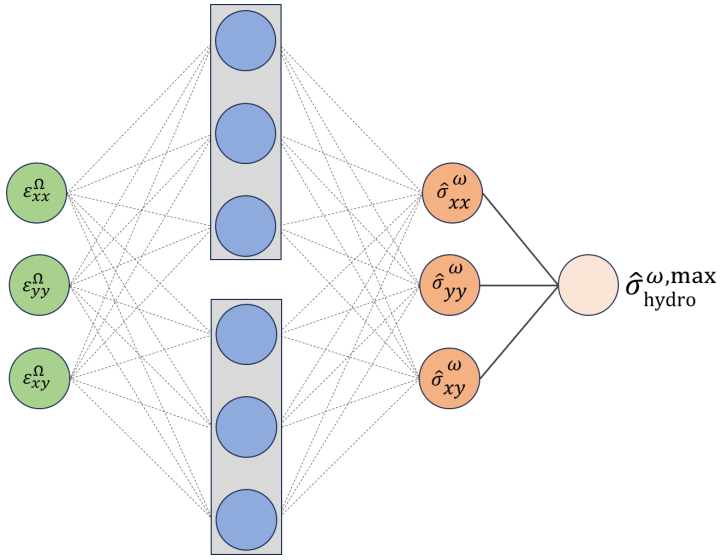
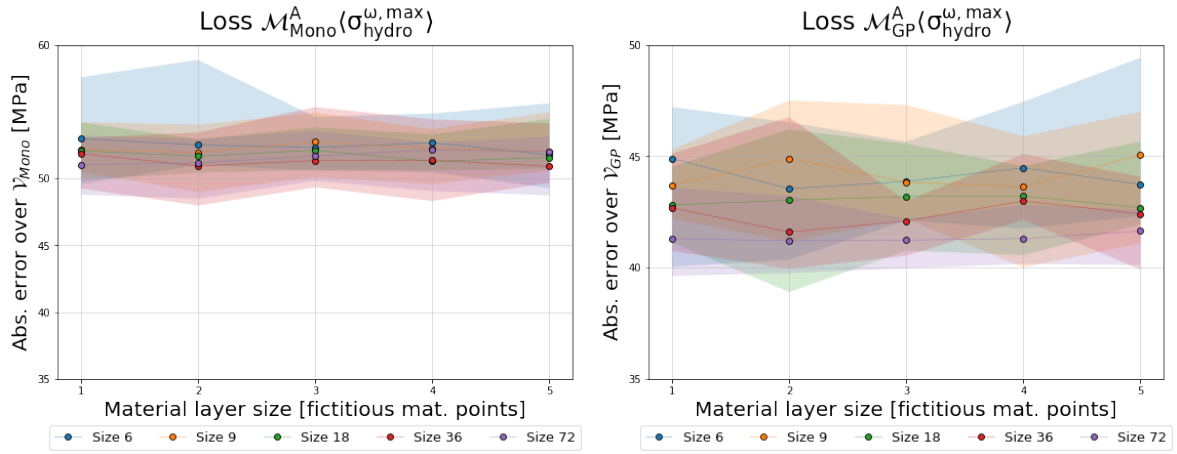


Figure 4.1: A new proposed architecture where the hydrostatic stress is computed after the original PRNN.

As the output and the target are different than in prior work, the optimal material layer size is also subject to change. A model selection procedure is done, where the material layer size and training size are variables. For this model selection, the material layer size range is $1-5$ and the amount of training curves used are selected from the numbers $n = 6, 9, 18, 36, 72$. The curves used during training are randomly sampled from the whole dataset. As the performance of the network is reliant on the sampled training curves and weight initialization, 10 networks will be trained for each combination of material layer size and training curve size. This is especially important for the combinations where the number of training curves is limited (e.g. $n = 6, 9$). The goal of the model selection is to find the most optimal number of material points for this network.

The batch size is set to 9 and the maximum number of epochs is set to 2500. The parameter to stop the training early is set to 500 epochs. This means that the training will stop if the validation loss has not improved for 500 epochs. This speeds up the computation time in case the training stagnates. The proposed network is trained on both the monotonic and the GP dataset. Figure 4.2a shows the validation loss of the model trained on monotonic data ($\mathcal{M}^A_{Mono}$) over the monotonic validation set ν_{Mono}. Figure 4.2b shows the validation loss of the model trained on GP data ($\mathcal{M}^A_{GP}$) over the GP validation set ν_{GP}. The horizontal axis indicates the material layer size and the colors refer to the number of training curves used. The colored dots illustrates the mean of the validation loss of the 10 networks per combination. The colored hatch show the range between the minimum and maximum validation error of each combination.



(a) Model selection on monotonic dataset

(b) Model selection on GP dataset

Figure 4.2: The result of the model selection for the monotonic dataset and GP dataset for model A

The results of the model selection in Figure 4.2 show that the performance of both networks did not yield the expected results. Usually the validation error decreases as the number of training curves is higher. For all material layer sizes this is not the case as the validation error is stuck at ~ 50 MPa. The range of the stresses in the matrix over all timesteps is between -65 and 65 MPa.

For the network with material layer size 3 and 18 training curves, the training and validation curves are shown (Figure 4.3) for both the GP and the monotonic dataset. Some comments on these plots are the following:

- The training of both networks was concluded after only a few epochs due to the early stopping mechanism. The green line indicates the moment that the validation error did not improve for 500 epochs.
- The validation curve of the monotonic dataset increases sharply after 100 epochs which indicates severe overfitting. This means the model learns the training data too well at the cost of its ability to generalize to unseen data.
- The validation curve of the GP dataset stabilizes after a few epochs, suggesting that the model has reached its best performance early on and does not improve afterwards.
- The training curves of both datasets show that the model is improving over the number of epochs, suggesting that the model learns from the data. The curve of the first plot shows additional learning over all epochs while the second curve stagnates after 200 epochs.
- The high validation error and signs of overfitting indicate that either the data is too complex, or the proposed architecture is not well suited for this task.

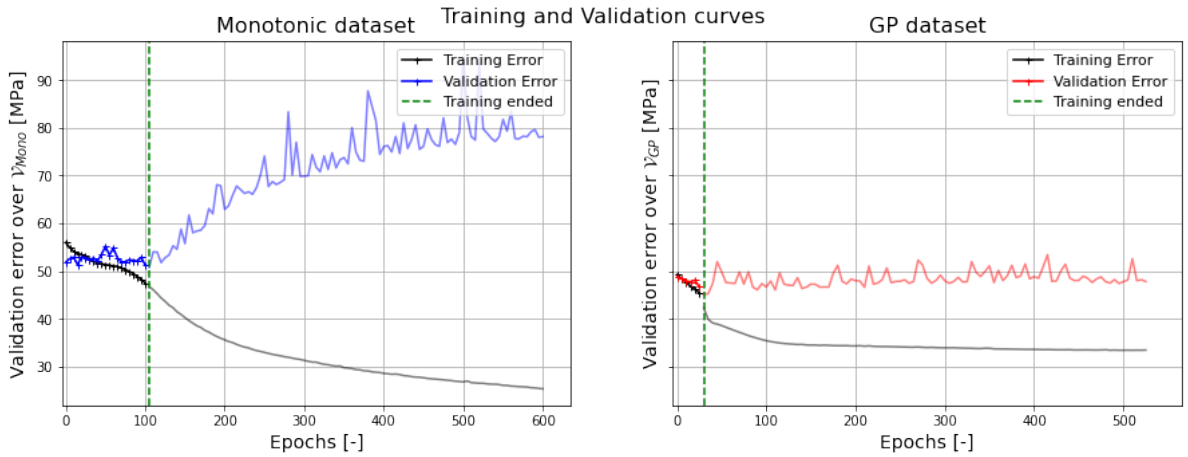


Figure 4.3: Validation and training curves of networks trained on GP and monotonic data for model A

So far, the training of the network and the resulting errors are underwhelming. Predictions made by the network are now analysed to see why this is the case. Figure 4.4 shows the results of one network trained on GP data that consists of 5 material points and was trained with 36 training curves. The first 5 figures plot 3 different curves. The dashed blue line is the computed hydrostatic stress from the 3 stress components in the material point to see what happens in the material point. The solid blue line indicates the decoded stress from an individual material point, or in other words the contribution of each material point to the final prediction. The prediction of the full network is shown in gray in each plot. The last subplot shows the prediction of the full network versus the target.

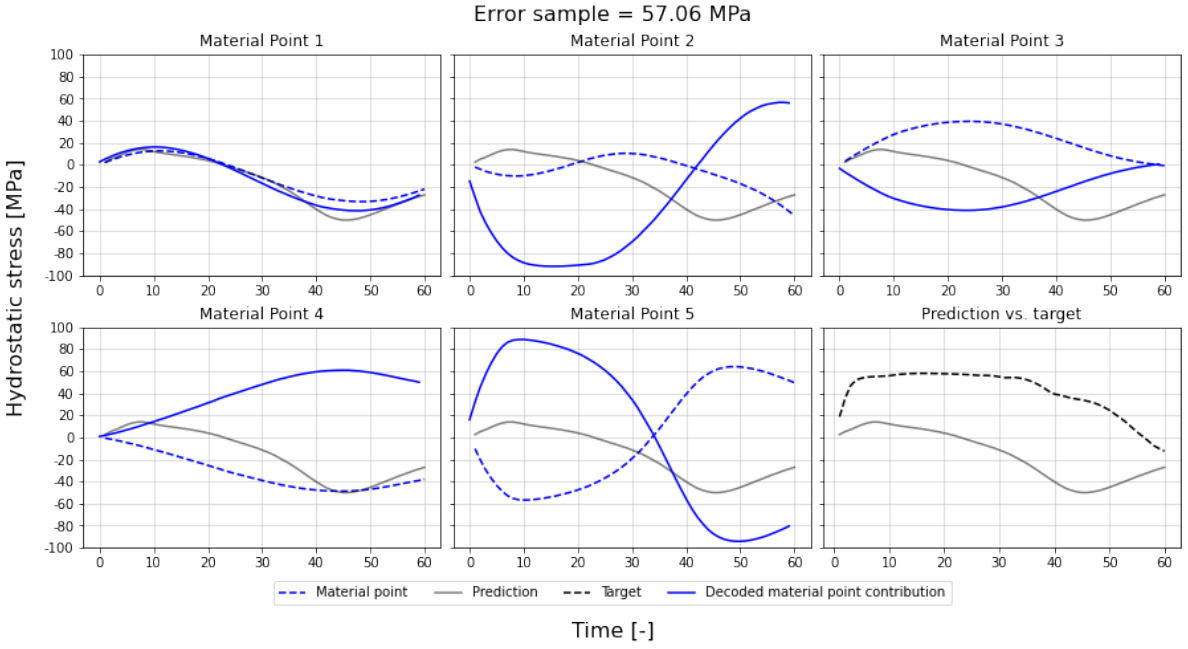


Figure 4.4: Results of the material points, the decoder and the total architecture for model A

As this network is overfitting on the training data and the validation loss is very high, drawing conclusions here is challenging. However, as the final prediction of the model is by the decoder that scales and adds up all contributions, this averages out the stresses with fixed contributions among the fictitious material points. This makes it very hard for this network to make accurate predictions to the maximum stress. Having the decoder made sense for predicting the average stresses, but clearly for this task it does not perform well.

4.2. Model B: Computing hydrostatic stresses before decoding

The first variant did not yield the expected results and so changes are made for the second variant. Even though it is suspected that the decoder does not favor this task, it is tried once more. Now, the hydrostatic stresses are computed inside the material layer. The reasoning is that this greatly reduces the number of trainable weights in the decoder. Model A has $9 \cdot m$ number of weights in the decoder while this model only has m number of weights in the decoder, with m being the number of material points in the layer. This simplifies the model and a smaller, simpler network is often easier to train and may converge faster and more reliably. Computing the hydrostatic stress is done after the J_2 material model generates the fictitious stresses and this is annotated with $\sigma_{hydro}^{J_2}$ in Figure 4.5. All other inputs are identical as model A in Section 4.1.

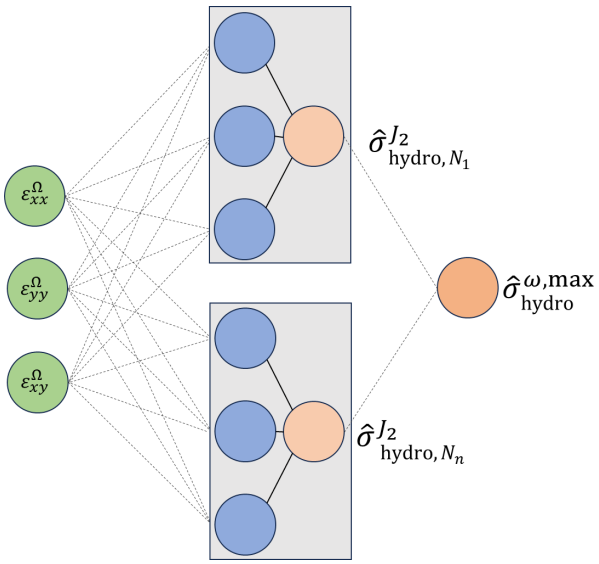
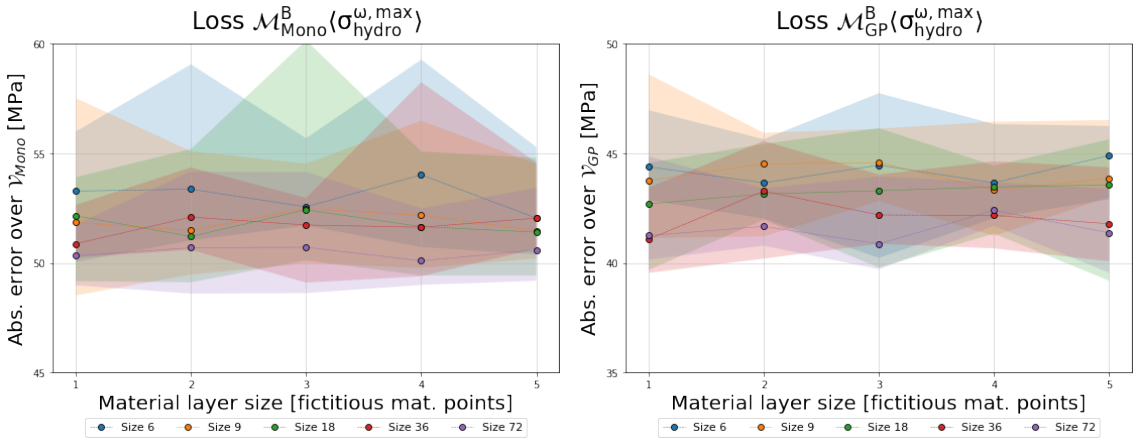


Figure 4.5: Results of the material points, the decoder and the total architecture for model B

Figure 4.6a shows the validation loss of the model trained on monotonic data ($\mathcal{M}_{Mono}^{B}$) over the monotonic validation set ν_{Mono}. Figure 4.6b shows the validation loss of the model trained on GP data ($\mathcal{M}_{GP}^{B}$) over the GP validation set ν_{GP}. The results are in line with the results from the first variant in Section 4.1 and thus no additional comments are made here.

The training and validation curves of two random samples are shown in Figure 4.7 for both the monotonic as the GP dataset. The results of these plots are in line with the results shown in Figure 4.3 of the previous variant. The final analysis for this model is to show predictions made by the network. So far, the results have been the same as model A which also has a decoder. No improvements have been made by changing the network by computing the hydrostatic stress in the material layer before decoding. Figure 4.8 shows the prediction of a network with 5 material points, trained on 36 training curves. The same conclusion can be drawn here. The decoder seems to average the stresses from the material points and thus the network is not able to predict maximum stresses. This gives reasons for the next model to have no decoder.



(a) Model selection on monotonic dataset

(b) Model selection on GP dataset

Figure 4.6: The result of the model selection for the monotonic dataset and GP dataset for model B

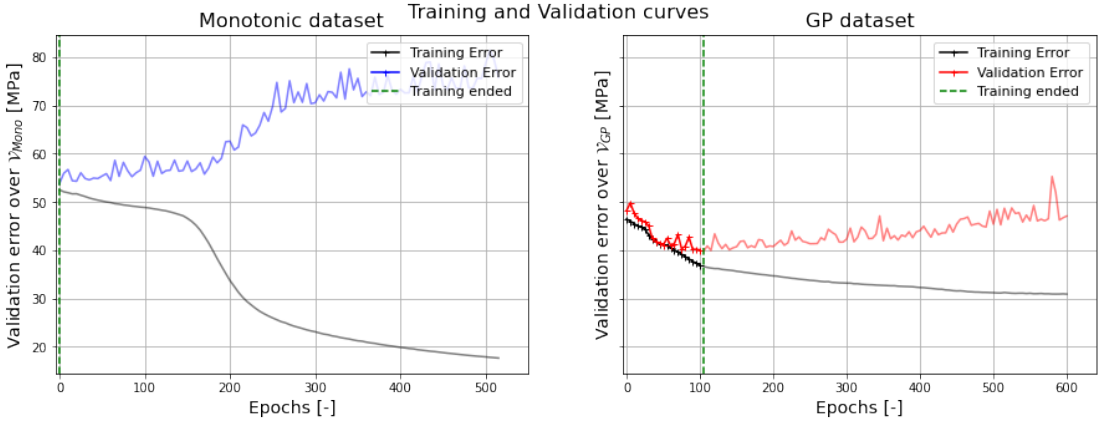


Figure 4.7: Validation and training curves of networks trained on GP and monotonic data for model B.

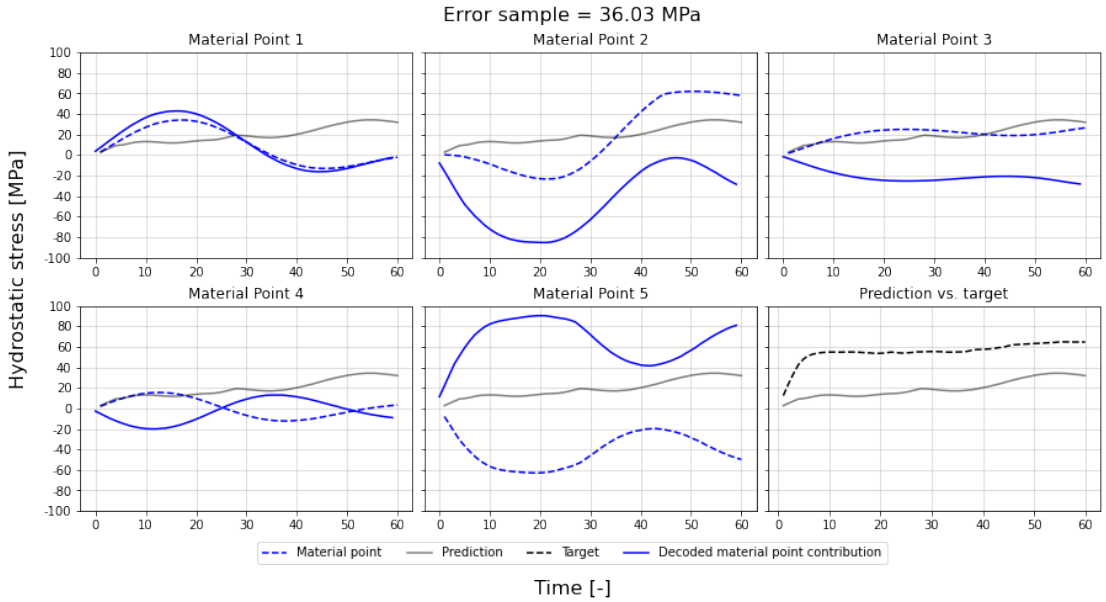


Figure 4.8: Results of the material points, the decoder and the total architecture for model B

4.3. Model C: Selecting the maximum stress without decoding

The first two variants in Section 4.1 and 4.2 did not yield the expected results. The decoder seems to average the stresses, which is not desired for finding the maximum. An architecture without a decoder is now considered. By removing the decoder, the learnable parameters in the network are reduced by half, which is beneficial during training as this takes less computational power. The network is less complex and is shown in Figure 4.9. The dashed lines are associated with weights and are trainable parameters, while the solid lines relate to functions like the computation of the hydrostatic stress from its components inside the material layer and selecting the maximum stress to compute the maximum hydrostatic stress.

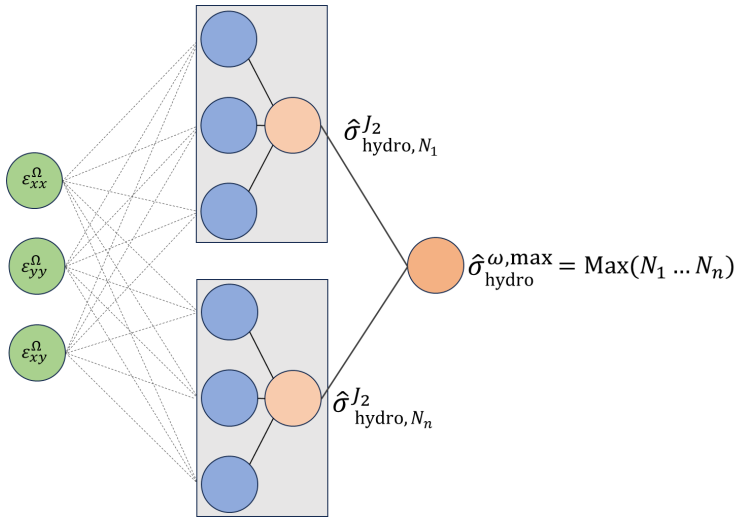


Figure 4.9: Proposed architecture with selecting the maximum stress after the material layer

The same settings are used from previous architectures like the number of epochs and number of training curves. Two changes are made. The first is the addition of a function that finds the maximum stress from all hydrostatic stresses that are computed inside the material layer. The second change is the increase in number of fictitious material points used in the material layer. The idea behind this is that there are more stress values computed by the material layer, increasing the chance that the maximum value is close to the target. The material layer size still uses $1-5$, with an addition of $7, 9, 11, 13, 15$.

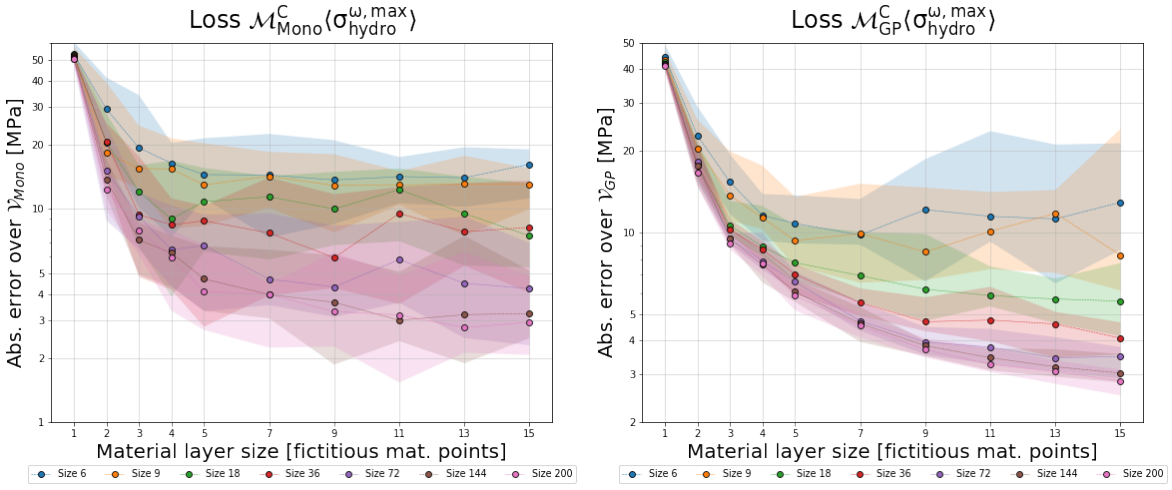


Figure 4.10: The result of the model selection for the monotonic dataset and GP dataset for model C

The results of the model selection are presented in Figure 4.10. Note the vertical axes are plotted

in log-scale to highlight the differences in the lower stress ranges more clearly. In comparison to the results of the previous architectures, these results are more in line with the expectation. The larger the number of training curves, the lower the validation error. The validation error also decreases when the material layer size increases, up to a moment when there are enough points. As this exact moment is impossible to predict, the model selection shows its worth as the plots presented clearly indicate the performance of the networks with varying parameters. The lowest absolute error over the validation set for this architecture is around 2 MPa, in line with the results achieved by the networks of Maia et al..

The training and validation curves are plotted for a random monotonic and GP sample respectively in Figure 4.11. The sample from both datasets is from a network with 5 fictitious material points and 36 training curves were used. The result on these curves also indicate a much better performance over the previous architectures. Some comments on these plots are the following:

- Both models show a consistent decreasing training error, with the model improving its performance on the training data. The training error of the monotonic dataset is a factor 2 lower than the GP dataset. This is because the variation in data in the monotonic dataset is much more limited than the GP dataset. The validation error for the monotonic dataset however is larger as some large outliers in the validation dataset influence the error.
- The validation error of the monotonic model decreases over the epochs as well, but the curve shows sharp fluctuations throughout the epochs. The sharp spikes, or noise, are due to mini-batches as the gradients for each weight update iteration will not be calculated on all available training data, but only a limited subset (batch). It seems unwanted but some noise is actually desirable as it may help the optimizer get away from local minima.

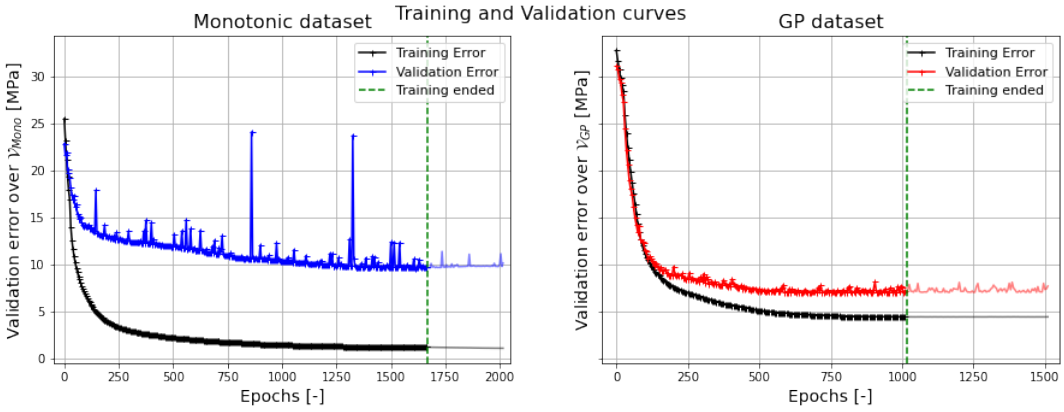


Figure 4.11: Validation and training curves of networks trained on GP and monotonic data for model C

Finally, a network with 7 material points is selected that was trained on the GP dataset with 36 training curves (Figure 4.12). Because this network has no decoder, there are only 2 curves in each plot. The blue dashed line indicate the hydrostatic stress in each material point. The gray line is the final prediction of the network. In the last plot, the target is shown in a black dashed line. The colors in this plot equal the colors in the other plots to see which material point is responsible for the prediction. What is interesting here, is that some points in this sample follow the maximum stress quite closely (material points 1, 3 and 7). Other points do not contribute to the final prediction at all. This does not mean that this architecture would perform better if only 5 material points were used though. Most samples that were investigated had some points not contributing to the overall prediction of the model. The error of this sample is only 3.72 MPa which is very impressive for a model with such a low amount of trainable parameters. This model has 7 material points (each consisting of 3 nodes) that are fully connected to the 3 input nodes. This results in 63 trainable weights. In the final plot of Figure 4.12 there are gaps between the predictions from different material points. This is because the solid colored lines connects consecutive predictions from a single material point between some timesteps. The gap indicates the next prediction is from a different material point.

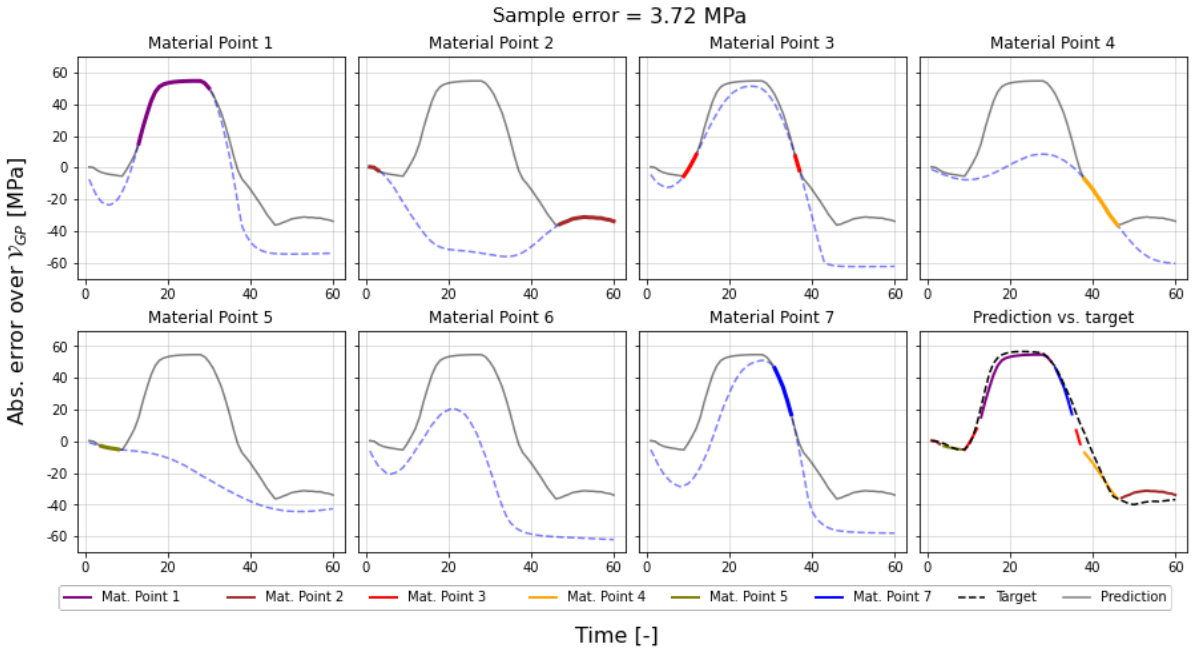


Figure 4.12: Results of the material points and the total architecture for model C with 7 material points

Figure 4.13 shows the predictions of a network with only 5 material points and trained on 6 curves. The error of this sample is just 4.13 MPa, showing the network is capable of using very little training data without the need for a lot of material points.

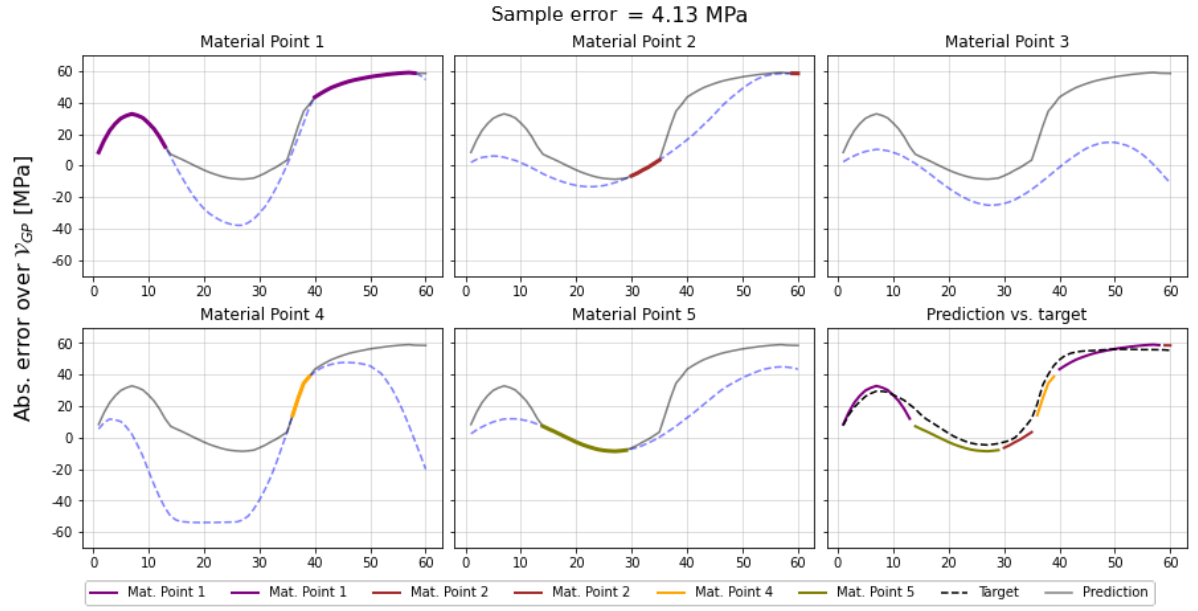


Figure 4.13: Results of the material points and the total architecture for model C with 5 material points

The idea behind this network follows the principle of finding the maximum stress in the RVE, by simply selecting the maximum occurring stress. When evaluating the stresses in the RVE, the maximum stress was often found in a different point. Having the freedom of changing points in this network works very well rather than using a decoder and have a stress contribution from each material point. The results of model C are excellent. This model has a very small number of trainable parameters ($9 \cdot m$ with m equals the number of material points), trains and converges very fast (< 2000 epochs) and shows to perform well on a few training curves. Therefore, from this point, only this network is considered. The

next sections in this chapter will show how robust this network is and the following chapters will show if this network can be leveraged to reach a higher potential in a multi-task approach.

4.4. Cross validation

Overfitting is one of the main concerns in machine learning. Cross validation provides a robust mechanism to detect overfitting. The architecture proposed in Section 4.3 showed promising results for the models that were validated with the same type of data. The model trained on monotonic data was validated with a monotonic validation set and the same holds true for the GP data. Additional analyses are done on this model to understand the performance in more detail. As each dataset may have unique properties, the cross validation helps to reveal how well the model adapts to these properties, while still generalizing well. In case a model trained and validated on the same data performs much worse when tested on another dataset, this means that the model does not generalize well to other datasets which may have different characteristics. In general, a model is more robust when it performs well on different datasets.

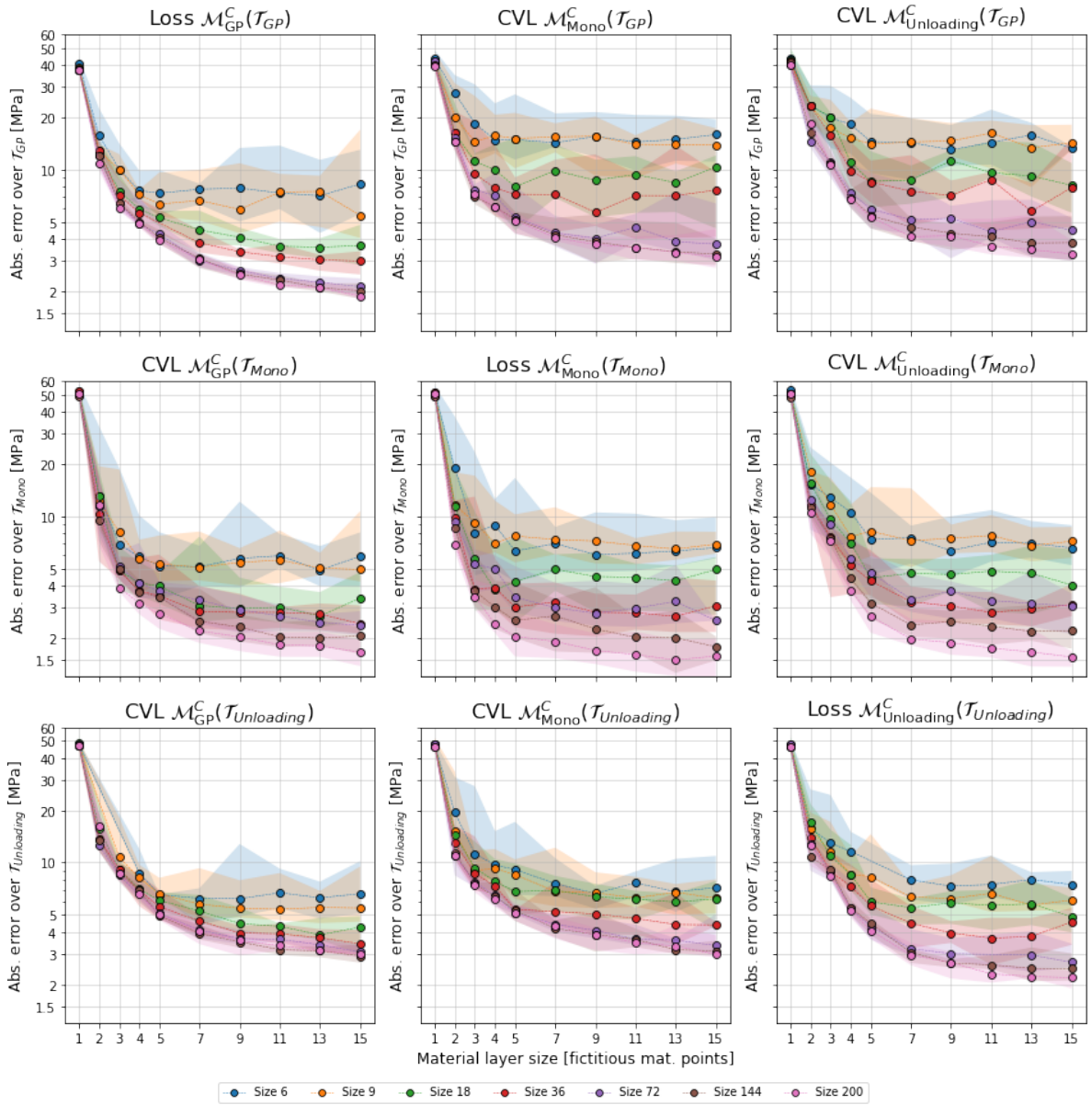


Figure 4.14: Cross validation between three test datasets for different training sets sizes and types considering model C

There are 3 different datasets in this work, which results in 9 analyses to complete the cross validation. Figure 4.14 shows the grid of plots. On the main diagonal, the same data is used for training and testing (e.g. $\mathcal{M}_{GP}^{C}(\mathcal{T}_{GP})$). The plots on the off-diagonal indicate the cross validation. The scale and the axes are the same for all plots. To check the robustness of a model, the columns of the grid show the different test sets of the same model. If one is interested in seeing which model to select when knowing the datatype a-priori, the rows should be scanned to see which model performs best. In the general case, however, we deem the GP-based curves as more general and better for model selection for applications where data-type is not know a priori. The following comments are made based on the cross validation:

- The first row shows the 3 models tested with a GP testset $\mathcal{T}_{GP}$. When comparing the three plots, one observation is that the model $\mathcal{M}_{GP}^{C}$ performs better than the other two as the error is about 50% smaller than the others. The complexity of the GP dataset is harder to capture by the monotonic and unloading model ($\mathcal{M}_{Mono}^{C}$ and $\mathcal{M}_{Unl}^{C}$).
- The second row consists of the 3 models that used the monotonic dataset as a testset. When comparing $\mathcal{M}_{GP}^{C}$ and $\mathcal{M}_{Mono}^{C}$, the range of errors appears to be more narrow. The range of for instance the material layer size 5 for $\mathcal{M}_{GP}^{C}$ is between $2.5 - 8$, while the error from $\mathcal{M}_{Mono}^{C}$ ranges from $1.6 - 18$. The models with a low number of training curves (indicated in orange and blue) also perform better with $\mathcal{M}_{GP}^{C}$ than for $\mathcal{M}_{Mono}^{C}$. Overall it is surprising that $\mathcal{M}_{GP}^{C}$ performs so well compared to $\mathcal{M}_{Mono}^{C}$, on which the dataset was originally trained on. When comparing $\mathcal{M}_{Mono}^{C}$ with $\mathcal{M}_{Unl}^{C}$ there are less obvious differences as the errors look very similar. The errors from both the $\mathcal{M}_{GP}^{C}$ and $\mathcal{M}_{Unl}^{C}$ with respect to $\mathcal{M}_{Mono}^{C}$ are interesting as they show a very good response to unseen loading, which shows flexibility in the model.
- The last row of the grid shows the performance of each model when tested on the unloading test set $\mathcal{T}_{Unl}$. It appears $\mathcal{M}_{GP}^{C}$ again shows a very narrow range of errors. The models trained on a smaller dataset perform better than $\mathcal{M}_{Unl}^{C}$, while the models trained on a larger dataset perform worse. The same holds true for $\mathcal{M}_{Mono}^{C}$, which is surprising as this model is not trained on a dataset with unloading.
- In general, the models on the main diagonal perform better than the models tested with another dataset. This suggests that the model adapted to specific characteristics of a dataset to a certain extent, which is expected. However, there are still plentiful of situations where the models on the off-diagonal perform better or similar than its equivalent on the main diagonal.
- The models can be ranked in the order $\mathcal{M}_{GP}^{C} > \mathcal{M}_{Unl}^{C} > \mathcal{M}_{Mono}^{C}$ when regarding their ability to generalize to unseen data types. This was expected as this is the same order of data complexity, however, this analyses was still proven very useful.

4.5. Stress-strain curves

Out of the three different architectures in Section 4.1, 4.2 and 4.3, the final architecture from Section 4.3 has proven to train fast and generalize well to unseen datasets. Furthermore, the absolute error is very impressive, even for models with a limited number of training curves. The model selection presented in Figure 4.10 is used to select the optimal material layer size. A suitable size is when the absolute error has reached its limit and does not decrease substantially when adding more material points. This is the case around size 11. To find out which amount of training data suffices, a fixed size test set $\mathcal{T}_{(\cdot)}$ of 100 samples is now used to plot the learning curves. This is done for all 3 datasets and the results are shown in Figure 4.15.

The x-axis indicates the number of training curves that were used during training and the y-axis shows the absolute error over the test set. When selecting the most applicable model from the training curves, one can again look at when the error has reached its limit. In this case, the models trained on GP or monotonic data show convergence when using 36 curves. The error of the model trained on the unloading dataset improves until 72 training curves were used.

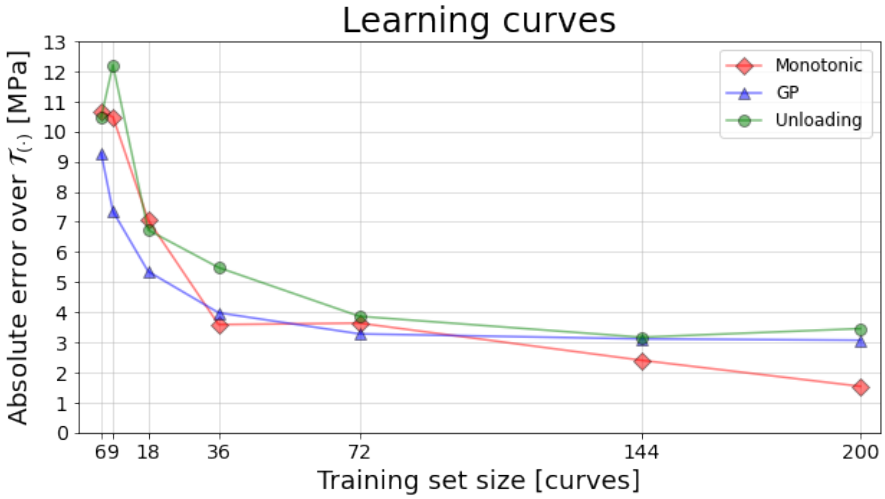
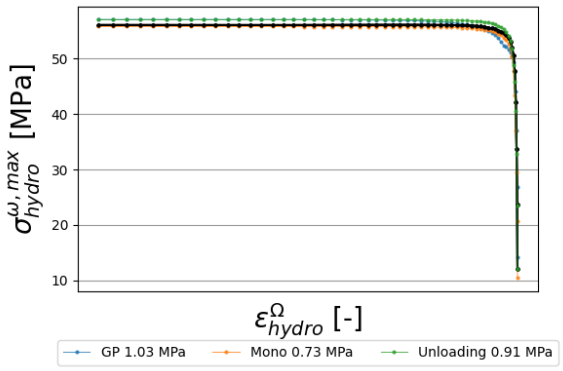
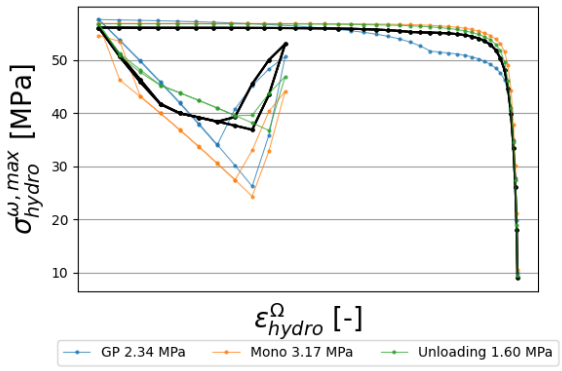


Figure 4.15: Learning curve for the best models with 11 material points and different training loading types

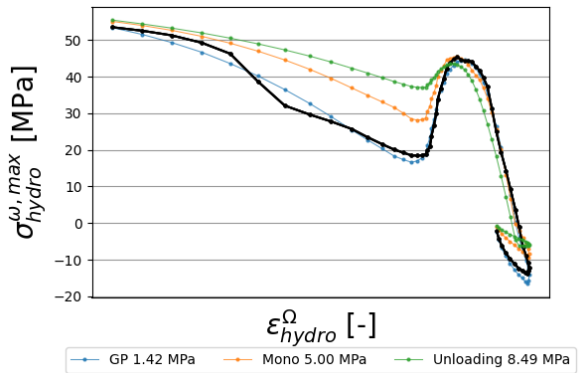
A few samples of each dataset are now ran through all 3 models to make predictions. Figure 4.16 shows the predictions made by all 3 models on a monotonic sample (Figure 4.16a), a sample with unloading (Figure 4.16b) and on a GP sample (Figure 4.16c). It was hard to find a sample that is representative for all 3 models at the same time. Overall, all models make very accurate predictions on the monotonic samples except for some very large outliers. The model trained on GP and unloading data performed well on the unloading sample while the model trained on monotonic data struggled. The GP sample was in most cases accurately predicted by the model trained on GP data while the other 2 models struggled.



(a) Monotonic sample



(b) Unloading sample



(c) GP sample

Figure 4.16: Stress-strain curves for a representative sample from a test set $\mathcal{T}_{(\cdot)}$, predicted by a model $\mathcal{M}_{(\cdot)}$

5 | Transfer Learning

Three different architectures were tested in Chapter 4 and the architecture without a decoder (Section 4.3) showed to perform best. Here, transfer learning to leverage knowledge from the original PRNN task which predicts macroscopic stress to the new PRNN design which predicts maximum microscopic hydrostatic stress will be tested considering only that architecture. The possibilities to apply transfer learning is limited because the material layer does not have trainable weights and so only the encoder can be used.

In both this work and the previous works, the model selection procedure has been applied numerous times, as it gives a detailed overview of the performance of a model with respect to training size and the number of fictitious material points that are used. The model selection by Maia et al. showed good performance for a model with 2 material points while the results in Section 4.3 show that more material points (11) are needed to predict the maximum hydrostatic stress accurately. The number of trainable weights in the encoder is dependent on the number of material points in the material layer. The input layer has a fixed size (3) and the material layer has $3\text{x}N$ nodes, where N is the number of fictitious material points. The encoder is fully connected, which implies that each node of the input layer will connect to each node of the material layer: $3\text{x}(3\text{x}N) = 9\text{x}N$.

This results in a shortcoming because the encoder of the average stress model has $9\text{x}2{=}18$ weights, while the encoder of the maximum hydrostatic stress model has $9\text{x}11{=}99$ weights. This discrepancy makes it challenging to apply transfer learning. It is possible to pass 18 weights from the former network but the remaining 81 weights that will be randomly initiated will still need training. For the first part of this chapter, transfer learning will be applied on models that have the same size. This means that the encoder of task 1 with N material points will be transferred to the model for task 2 that also has N material points. Section 5.4 explores a strategy where the number of material points is not the same for the 2 models.

After the weights from the former model are initiated in the second model there are two options. The weights can either be frozen during training or they are allowed to be optimized. Freezing weights can be beneficial to speed up the training process of the new model. In case the weights are frozen, there are less trainable parameters. Secondly, freezing a layer enforces the layer to perform the same task as it was trained on before. In this specific case, the first model has *"learned"* to transform fictitious strains to fictitious stresses. As this part of the computation will be the same for both models, freezing can be effective. Freezing the encoder will however lead to 0 trainable parameters when applied to the model without a decoder.

The second option is when the transferred weights are not frozen and can be optimized during training. This is referred to as a *warm-start*. Warm-starting is commonly used in machine learning. In case the training of a model has stopped, warm-starting is used to continue training. Another case is when new or higher quality data becomes available and the optimized weights of the prior model are often a good starting point. Warm-start can lead to faster convergence which reduces training cost. In this work, both warm-start and freezing the encoder will be tested.

For completeness of this analysis, the order of the tasks will also be taken into account when transferring the weights from an optimized model to another. The first analysis of Section 5.1 will show transfer learning from average stresses to maximum hydrostatic stresses and the second analysis is in reversed order: maximum hydrostatic stresses to average stresses.

5.1. Results direct training versus transfer learning

This section will first present the results for transfer learning from a task predicting the average macroscopic stress to a second model predicting maximum hydrostatic microscopic stresses. The second section will show the results for the order of the tasks switched.

Transfer learning: average to maximum stresses

In this part the results are presented for the models that predict the maximum hydrostatic microscopic stresses. Figure 5.1 shows the average results over 10 different models for each combination of material layer size and training set size. Each color, representing a training set size, is presented three times. The solid line with a circular marker shows a model directly training for maximum hydrostatic microscopic stresses and there is no transfer learning applied here. The dashed lines with a triangular marker shows the results of transfer learning with warm-start and the solid lines with a square marker indicate transfer learning with a frozen encoder.

In the case of transfer learning, first a model was trained predicting the average macroscopic stresses. Then, the encoder is copied and the second model is initialized that inherits the encoder. Finally, the training is then started with either a frozen encoder or with warm-start. The same training set is used in the case of transfer learning for model 1 and model 2.

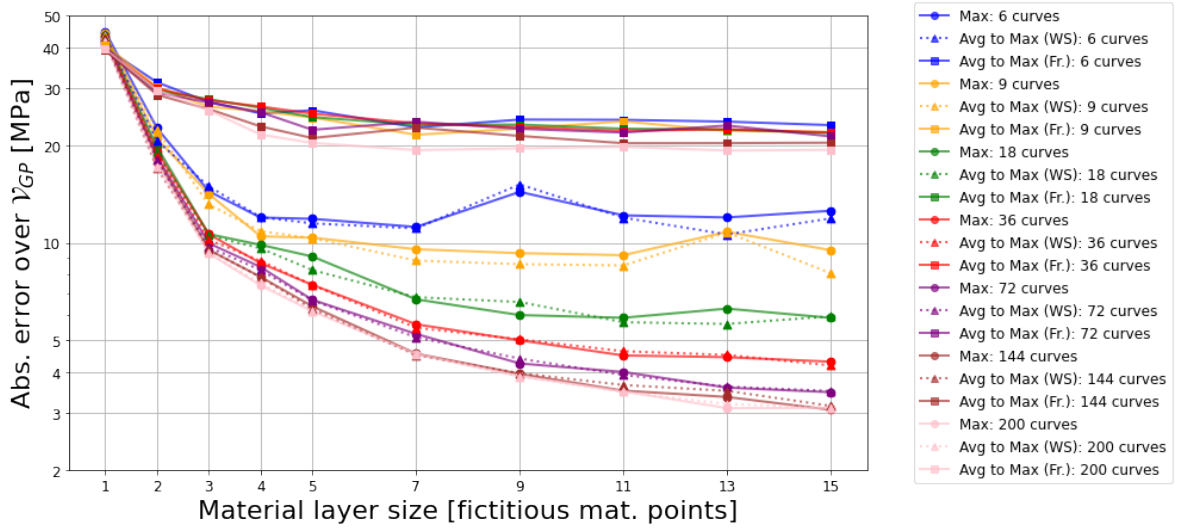


Figure 5.1: Transfer learning (Avg to max) versus direct learning for maximum stresses, results are the average results over 10 networks for each combination for GP data

The results shown in Figure 5.1 are the average results over 10 networks and indicate no clear difference between direct training versus transfer learning with warm-start. In case the encoder is frozen, no training is done and the results show that the predictions for the second task are not accurate. Figure 5.2 shows the best model for each combination and the same conclusions can be drawn here. Furthermore, the results remain unchanged regarding the fact that models with more points and or more training curves perform better.

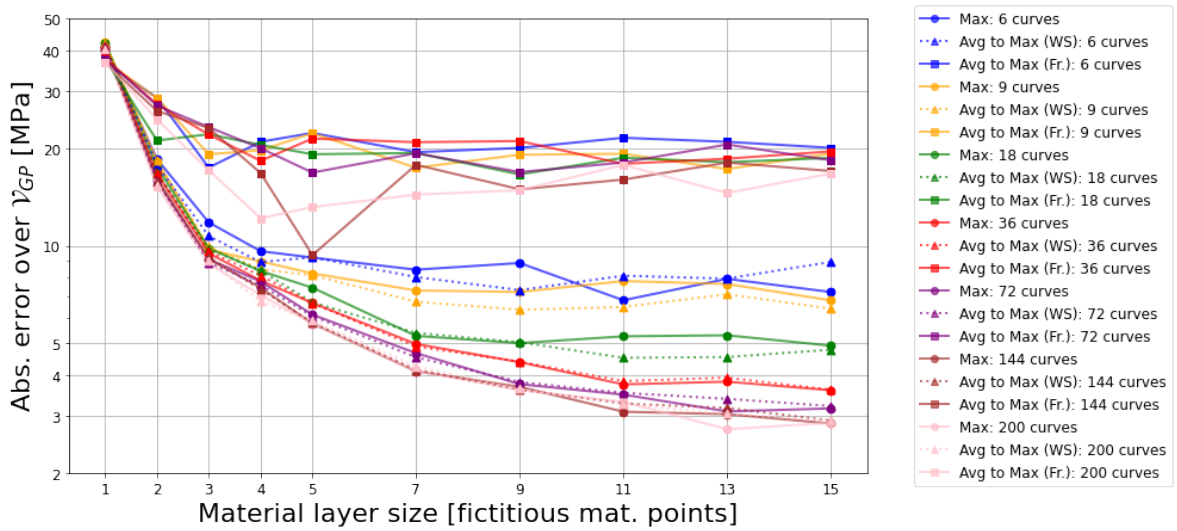


Figure 5.2: Transfer learning (Avg to max) versus direct learning for maximum stresses, results show the minimum error over 10 networks for each combination for GP data

Transfer learning: maximum to average stresses

This section will present the results on transfer learning versus direct training in case the tasks are swapped. First, a model is training to predict the maximum hydrostatic microscopic stress. Then, a new model is initialized which inherits the encoder. In this case however, the new model has a trainable decoder. Warm-start and freezing the encoder are then used to train the model and the results are presented here. The same training set is used in the case of transfer learning for model 1 and model 2.

Figure 5.3 shows the average of 10 networks for the direct training versus transfer learning. First of all, the general trend is that for each unique number of training curves, there seems to be a specific number of material points from which increasing this number is no longer beneficial. The blue and yellow curves indicate that only 2 points are needed, while models with more material points need more training curves to learn the higher number of parameters. This behavior is the same for direct training and transfer learning.

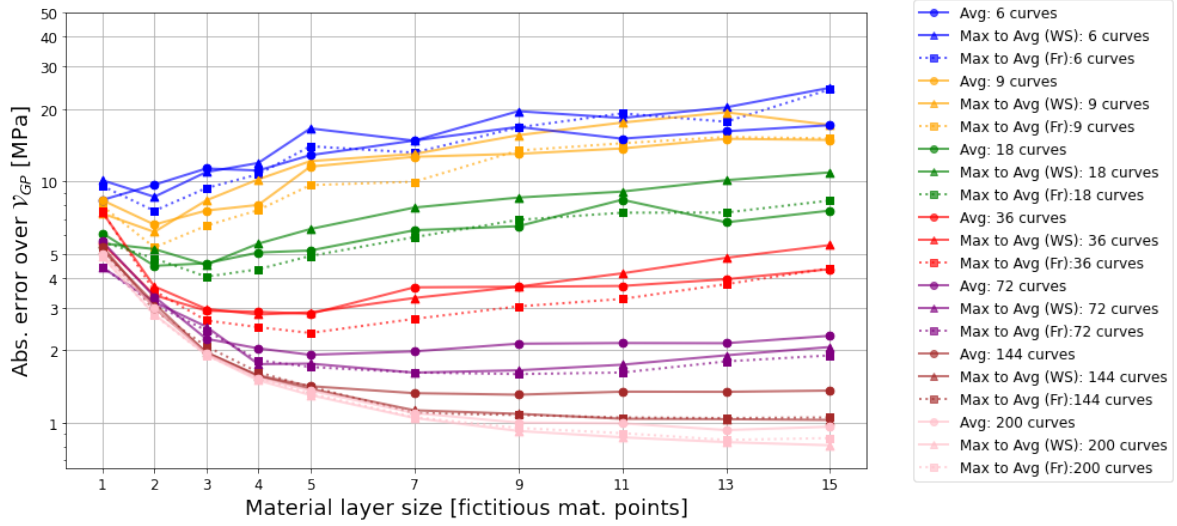


Figure 5.3: Transfer learning (Max to Avg) versus direct learning for average stresses, results are the average results over 10 networks for each combination for GP data

When one compares the direct training versus transfer learning, it seems that the models that applies transfer learning with a frozen encoder perform slightly better than the other 2, as the dashed line is below the 2 solid lines for most instances. However, differences are small and no clear winner can be stated. The same results can be drawn when analyzing the best models which are presented in Figure 5.4.

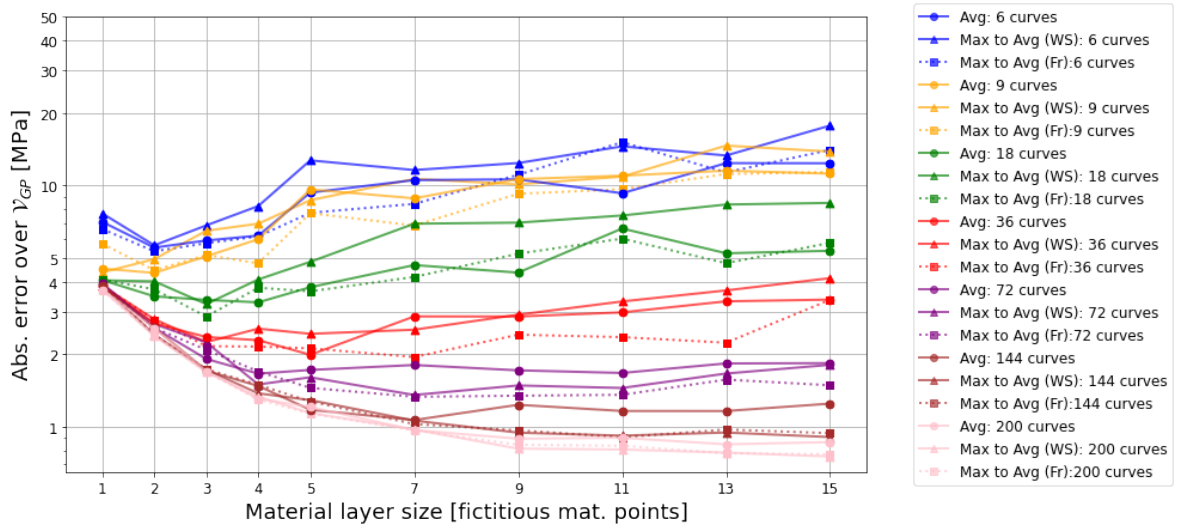


Figure 5.4: Transfer learning (Max to Avg) versus direct learning for average stresses, results show the best model over 10 networks for each combination for GP data

The results of these 2 subsections indicate no clear winner when comparing direct training versus transfer learning. For some cases, the model that applies transfer learning with a frozen encoder seems to perform slightly better. This may have to do with the fact that there is less trainable parameters and the model can only focus on tuning the decoder. Prior to this analysis, the idea was that the frozen encoder would possibly lead to a model that was too rigid, but this is not the case.

5.2. Training performance

The results from Section 5.1 showed that transfer learning performs equal or worse regarding absolute error compared to models that are trained with a random initialization of weights. In this section the training performance is investigated by plotting the validation curves. The introduction of this chapter explained that training can be faster for transfer learning and this can only be seen by plotting the validation curves.

The results in the previous section showed that most models were still improving the validation loss. For this reason the number of epochs is increased to 5000. For this analysis the transfer learning is only shown for average to maximum stresses. The early stopping feature was set to 500 to stop in case the decrease in validation loss stopped. The circular markers indicate when the training of a model is concluded. Figure 5.5 shows the validation curves of the 2 different models. The red curves and box plots indicate the direct training of the maximum hydrostatic stress without transfer learning. The blue curves are the models where transfer learning is applied. Warm-start is applied to ensure training the encoder. Both the GP and monotonic dataset are used. The same sampled datasets were used for the model with and without transfer learning in order to be able to compare the models.

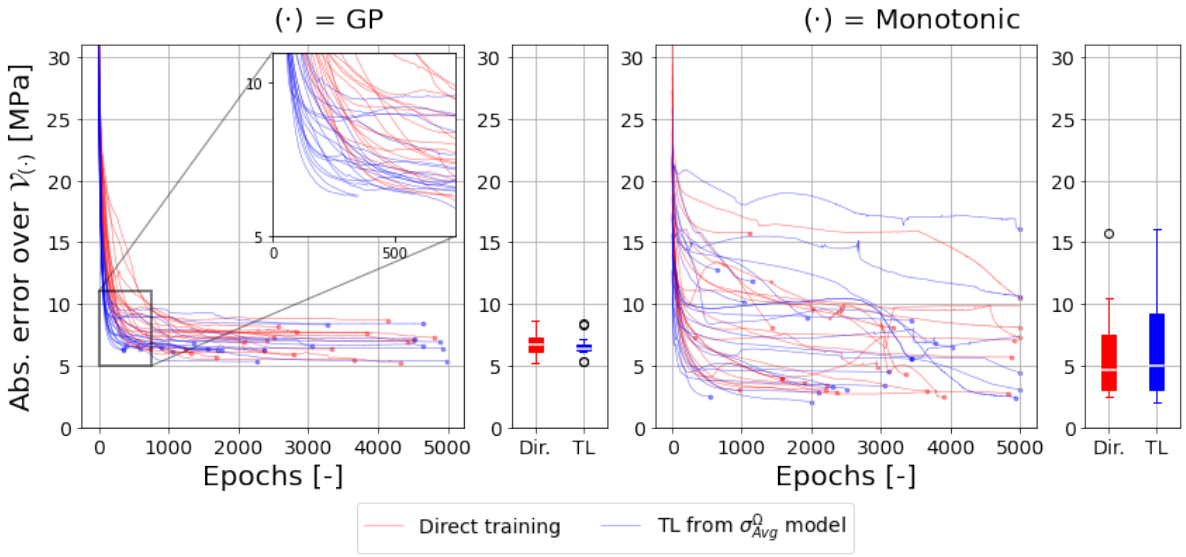


Figure 5.5: Comparing regular training with transfer learning for GP and monotonic data

The following observations are made:

- The models with transfer learning in the first plot ($\mathcal{M}_{GP}$) are converging faster. The secondary figure inside the first plot shows the area between 0-750 epochs more clearly. This shows a better performance for the models where transfer learning was applied. Most of the blue validation curves have a lower error than the red curves.
- For this computation, 50% of the models with transfer learning had converged to a minimum before 2000 epochs. This was the case for only 25% of the models without transfer learning. This can be seen by the red and blue circular markers that indicate when training was concluded.
- When looking at the first set of boxplots in Figure 5.5 for $\mathcal{M}_{GP}$, it appears that the models with transfer learning perform slightly better than the models without transfer learning. The median of the 20 models is a fraction lower and the interquartile range (IQR) is more narrow than the model trained from scratch.
- The training curves from the models trained on monotonic data $\mathcal{M}_{Mono}$ do not show a consistent better performance for transfer learning. The same instability is visible in these validation curves as during the model selection of this architecture in Section 4.3.
- The box plot for the monotonic dataset show quite similar performance between the networks. The IQR of the networks without transfer learning is more narrow while the best model with transfer learning have a slightly lower validation error.

5.3. Selecting the encoder

The final analysis that is done in this chapter is to see if the best performing encoder from task 1 will result in the best performing encoder for task 2. So far 20 networks were initiated to directly train for average stresses $\mathcal{M}(\sigma_{Avg})$. This resulted in 20 different sets of encoder weights. These 20 sets were then initialized to warm-start the training for the second task: predicting maximum hydrostatic stresses $\mathcal{M}(\sigma_{Max})$. In Figure 5.6 the comparison is made visible by linking the 20 models. Each line indicates the set of weights that is transferred with transfer learning. Two lines are highlighted. The red line indicates the best performing model predicting average stresses $\mathcal{M}(\sigma_{Avg})$, linked to the resulting model predicting maximum stresses using the weights of the first model. The blue line shows the best performing model with transfer learning $\mathcal{M}^{TL}(\sigma_{Max})$. The analysis is done both for the GP as the monotonic dataset.

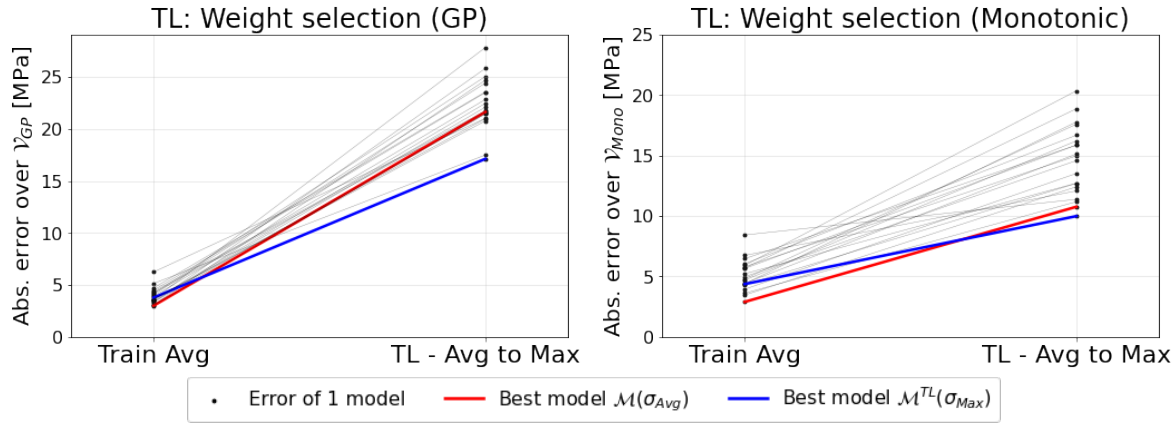


Figure 5.6: Result of the analysis if the lowest error model of task 1 results in the lowest error for task 2

The results show that the best performing model from task 1 does not guarantee the best results for task 2. This would be the case if the red line would connect the two lowest markers. This result is a setback because ultimately the goal is to have a model with the lowest error possible. Most connecting lines do appear to run parallel to each other suggesting that the models are often ranked in the same order but this is not a given for all.

5.4. Different architecture sizes

As briefly mentioned in the methodology, one challenge is when the architecture size is different between 2 models. When training a model predicting average stresses, only 2 or 3 material points are needed to have a well performing model. In the case of a model predicting the maximum stress, more points are needed. In this section, the encoder of a model predicting average stresses is taken. This model has 2 material points and a weight matrix of size $[6, 3]$. Then, transfer learning will be applied to train a model predicting maximum stresses. The material points for this second model is the same as used so far $(1 - 5, 7, 9, 11, 13, 15)$.

To deal with the change in weight matrix size, the weight matrix of model 1 is first copied to the second weight matrix. Then, additional weights of model 1 are sampled to the second weight matrix. Sampling is done per column. This is done to keep the weights in the same position regarding its axis (a weight that was used in x-axis will stay in the x-axis). An example of this process is shown in Figure 5.7. The first part shows an example weight matrix of task 1. In this case model 2 has 6 material points which results in the weight matrix of the encoder to be of size $[18, 3]$. In the second window an array filled with zeros is initialized and the weights of model 1 are copied in the first slots. Then the remaining rows are sampled from the original model. The numbers used here are arbitrary.

```
Weight matrix model 1:

 [[ 1  2 -4]
 [ 4 -3 -1]
 [-1  1 -3]
 [ 4 -5  0]
 [ 3  3  4]
 [ 3  1 -5]]
```

```
Copied weights of model 1:

 [[ 1.  2. -4.]
 [ 4. -3. -1.]
 [-1.  1. -3.]
 [ 4. -5.  0.]
 [ 3.  3.  4.]
 [ 3.  1. -5.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
```

```
Sampling the remaining weights
from model 1 per column:
 [[ 1.  2. -4.]
 [ 4. -3. -1.]
 [-1.  1. -3.]
 [ 4. -5.  0.]
 [ 3.  3.  4.]
 [ 3.  1. -5.]
 [ 4.  2. -5.]
 [ 1. -3. -4.]
 [ 1.  2. -5.]
 [ 1.  1. -4.]
 [ 4.  1. -5.]
 [ 1. -3. -1.]
 [ 4. -5. -1.]
 [ 4.  2. -4.]
 [ 4.  1. -4.]
 [ 1.  1. -5.]
 [ 3. -5. -5.]
 [ 1.  3.  0.]]
```

Figure 5.7: Dealing with different layer sizes by copying and sampling the original matrix

This analysis is only done for transfer learning from the average to the maximum stress. This is because the network predicting average stresses needs less points than the network predicting maximum stresses. The results are shown in Figure 5.8 and indicate the best model of 10 networks for each combination. The circular markers with the labels 'Matching Task 1' indicate that the weights are sampled from a smaller encoder. The triangular markers with the labels 'Fixed Task 1' indicate that the material layer size is fixed in both networks. The results are comparable to the network where the layer size of the model of task 1 is the same as task 2.

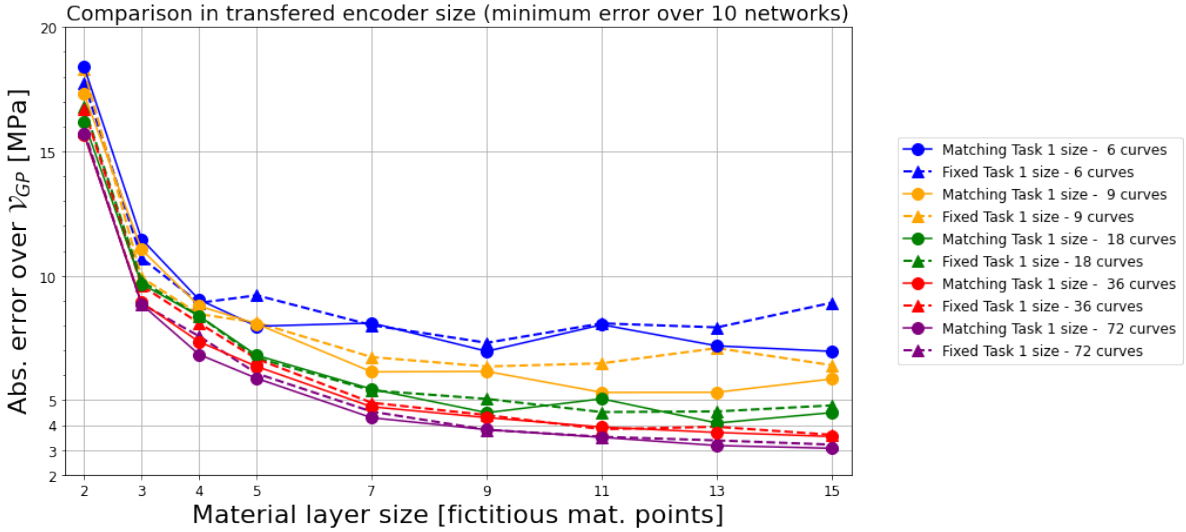


Figure 5.8: Comparing transfer learning models with equal and unequal material layer size for GP data

Finally, the validation curves of the models with a different inherited encoder are checked. The results are shown in Figure 5.9. For this analysis, 7 material points and 18 training curves were used and this is compared to the validation curves of a model with 2 material points and 18 training curves as well. The first part of the figure indicates the validation curves for the model predicting the average task. The blue curves are from the model with only 2 points and it is already known that this model performs better than the model with 7 points when predicting the average stress. Each model is initialized 10 times and trained with a different sampled dataset.

Then the weights of the encoder are used to train the second model. The encoder from the model with 7 points are used to initialize the second model without modifying the encoder. The encoder for the model with only 2 points are now modified by the sampling procedure. There is no clear evidence that the model with a modified encoder performs different than a model that inherits weights that are not optimized for task 1. Note that the validation curves in Figure 5.9 are smoothened as the noise made the curves hard to identify.

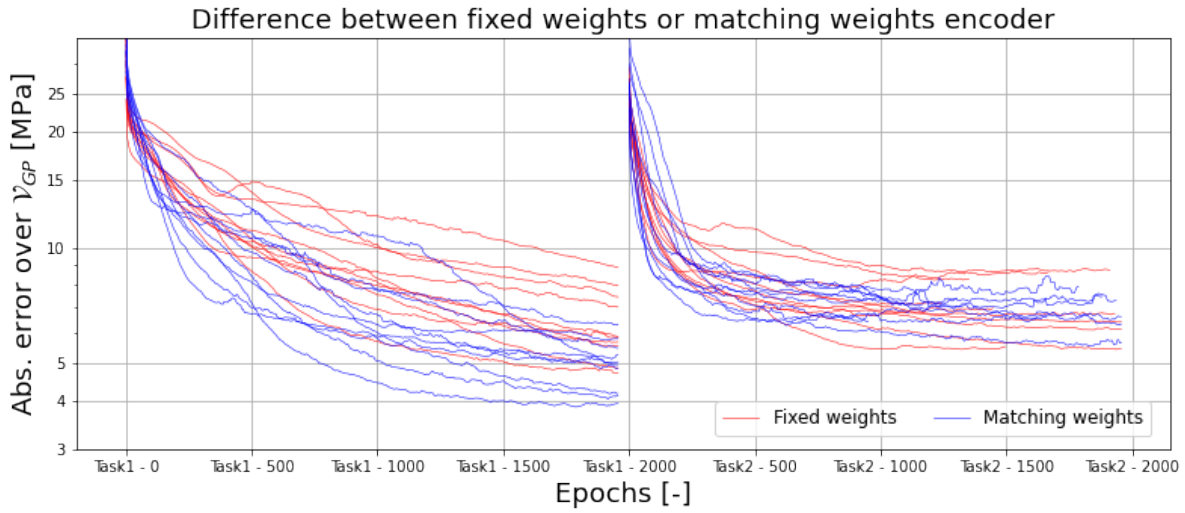


Figure 5.9: Comparing validation curves for models with equal and unequal material layer size

The main takeaway from this final section is that at least for these two tasks it does not matter if the encoder that is inherited by task 2 is not optimal for task 1. The validation curves show no faster learning or better convergence than with optimized weights from model 1.

6 | A multi-tasks approach

The previous chapter has shown that the encoder with the material layer of the original PRNN also works well for predicting the maximum hydrostatic stresses. Because the architecture in predicting the maximum did not result in more trainable parameters, the idea of combining two tasks in a single model arose. This multi-task approach is discussed in this chapter.

6.1. Combining tasks

As briefly mentioned in the introduction of this chapter, there are no additional trainable parameters when combining the two tasks in one model. Both models use the same fully connected encoder with linear activation functions and a material layer. The output layer of the average stresses includes a fully connected decoder which has trainable parameters, while the output layer of the maximum stresses only picks the maximum value from the material models. Figure 6.1 shows how the stresses in the material layer are passed to both output layers. The loss function now has to account for both quantities. The starting point to combine the losses is simply to add both losses in the loss function.

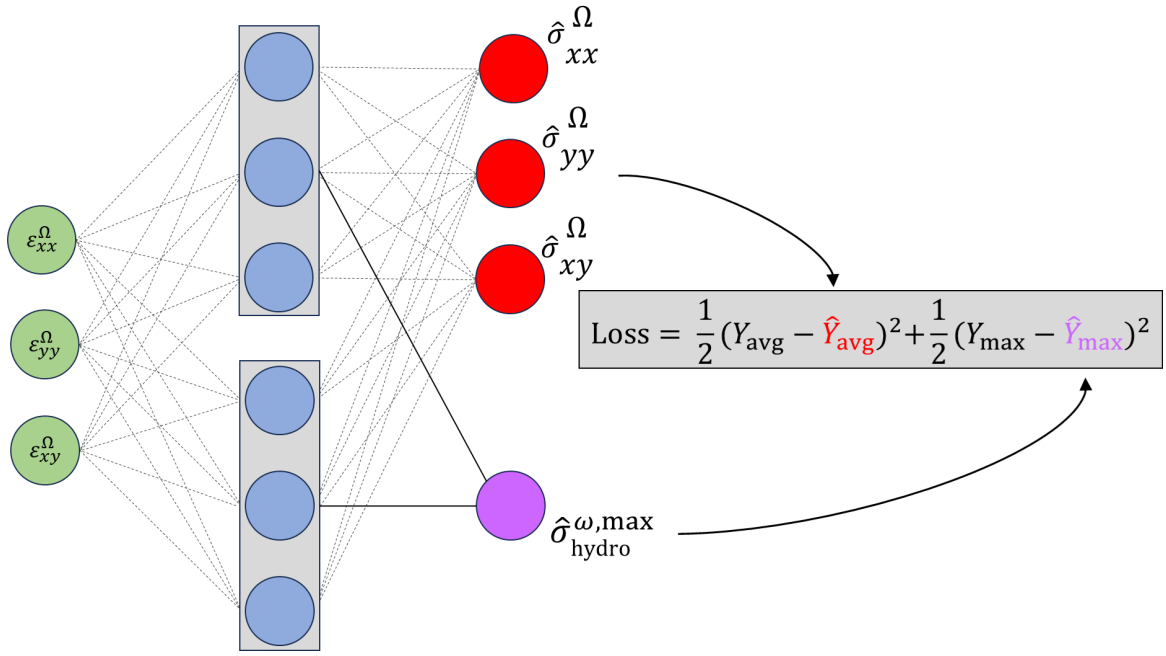


Figure 6.1: The architecture of the combined model, computing both average and maximum stresses

The model selection procedure is done for this network as well and the results are shown in Figure 6.2. The same parameters are used as the previous model selection. The magnitude of the absolute error on the validation error cannot be compared to the errors found in previous chapters as it now consists of 2 losses. The following comments are made on the results:

- The first model in Figure 6.2a show that the models improve as the material layer sizes are increased from $1-5$. From that layer size, the validation error does not improve, or gets worse for the models with a limited number of training samples (6, 9 and 18). For the models with training size 18 the validation error does not improve after the material size is increased to 7 before stabilizing. The models with a larger training set still show notable improvements as training size and material layer size increase.

- As each combination of training size and material layer size is trained on 10 different networks with a different training set, the validation errors have different outcomes and this is indicated by the colored range in the plots. There is a noticeable increase in validation error range for the models with a small training set as opposed to the models with a larger training set. It is advantageous that the models with a large training set have a much more narrow range of validation error.
- The range of validation error is much higher for $\mathcal{M}_{Mono}$, also for larger training set sizes and material layer sizes. This is mostly due to outliers that negatively influence the results.

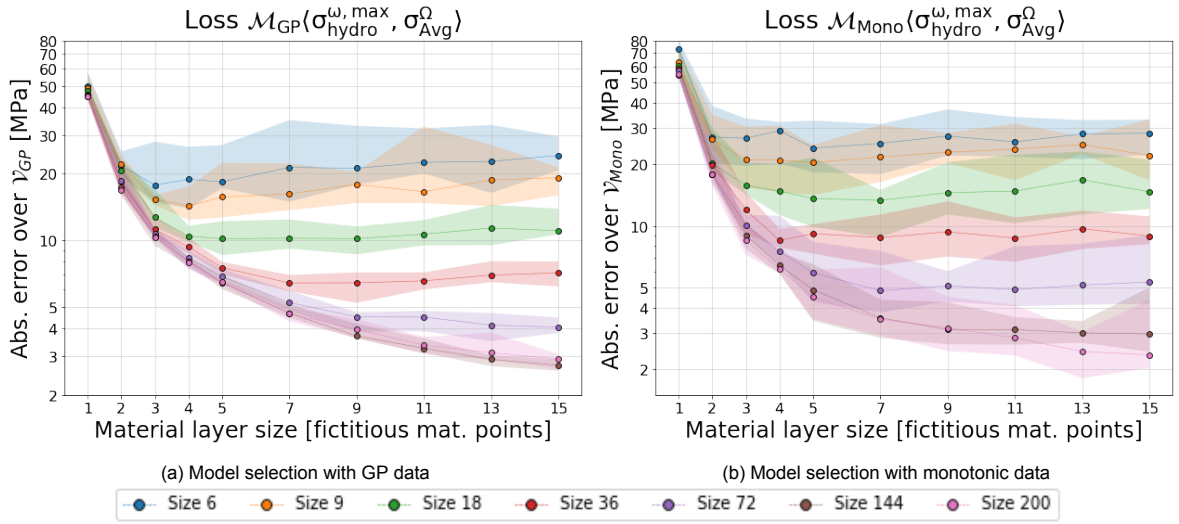


Figure 6.2: Model selection with combined model that computes both average and maximum stresses

6.2. Comparing single and multi-task models

In this section the results of the multi-task model $\mathcal{M}_{GP}(\sigma_{Max}, \sigma_{Avg})$ of section 6.1 is compared to the results of the single-task models $\mathcal{M}_{GP}(\sigma_{Avg})$ and $\mathcal{M}_{GP}(\sigma_{Max})$. The loss function of the multi-task model is the addition of the two losses and so the validation losses of the single-task models are summed as well. This summation is done on both the average and the best of the validation error of 10 models per combination of material layer size and training set size.

Figure 6.3 shows the results for different number of material points, training set sizes and types. Only the average validation error is shown in Figure 6.3 and not the full range of errors. This is done to be able to compare the results more easily as the colored hatches would overlap. The triangular markers indicate the multi-task model and the circular markers represent the validation error for the combined single-task models.

When one compares the single- and multi-task models it becomes evident that on average the multi-task models perform better for nearly all combinations of material layer size and training size. The early expectations were that if one is to model both average stresses and maximum microscopic stresses, it would be best to have 2 models trained for 2 tasks. The performance of the multi-task model is remarkable as it performs better than the combined loss of 2 single-task models and thus exceeding the expectations.

For completeness, the minimum validation errors of the single-task models are also combined. These are compared to the best performing multi-task models. The results are shown in Figure 6.4. The results for the GP dataset are in line with the average validation error: the multi-task model show a better performance for most combinations of material layer size and training set size. The best performing single-task models trained on the monotonic dataset do perform better than the multi-task model for a low number of material points. This is because the models trained on the monotonic dataset in general have low errors, but very large outliers. These outliers are filtered and so the result favors the 2

single-task models for a low number of material points.

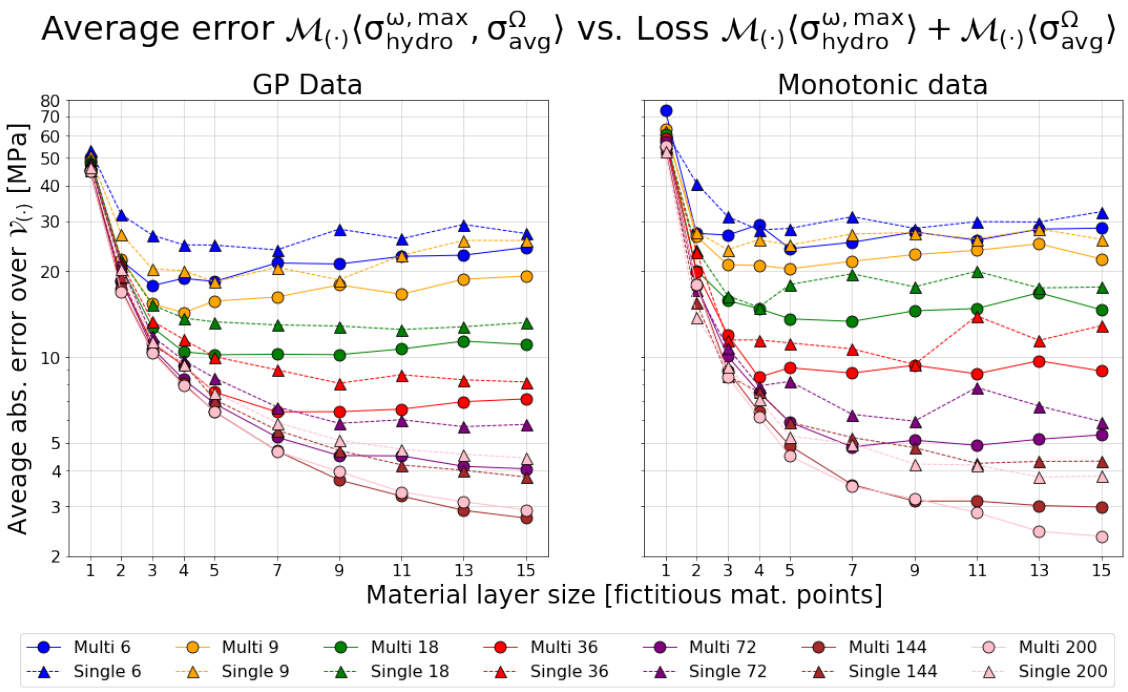


Figure 6.3: The comparison between the combined single-task and multi-task average validation error

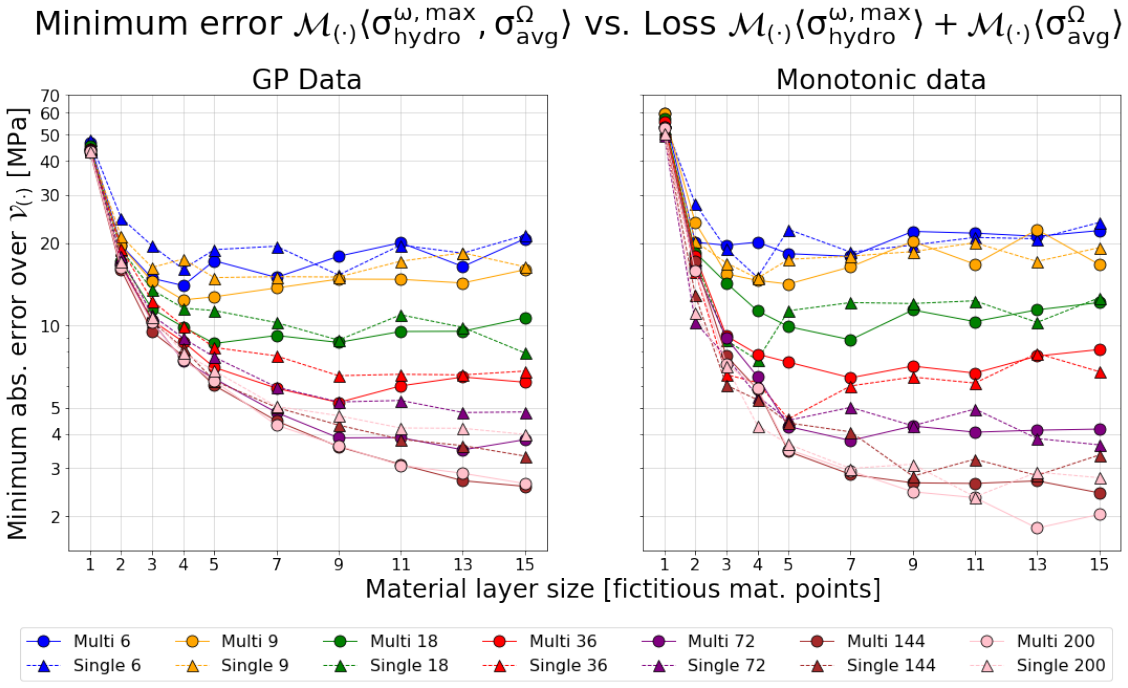


Figure 6.4: The comparison between the combined single-task and multi-task minimum validation error

In the 2 figures above only the total error is shown for the 2 single and the multi-task approach. All individual loss components are known and can be plotted as well. This gives more insight in the performance of the models. This is done in Figure 6.5 for the average error over 10 networks for each combination for the GP dataset and the minimum error for models trained on the monotonic dataset

is shown in Figure 6.6. The blue solid lines with the circular markers indicate the total loss of the 2 individual models. This total loss consists of the average macroscopic loss, shown in a solid blue line and the maximum hydroscopic stress shown in the blue dashed line. The results for the multi-task is shown in the same layout but in red.

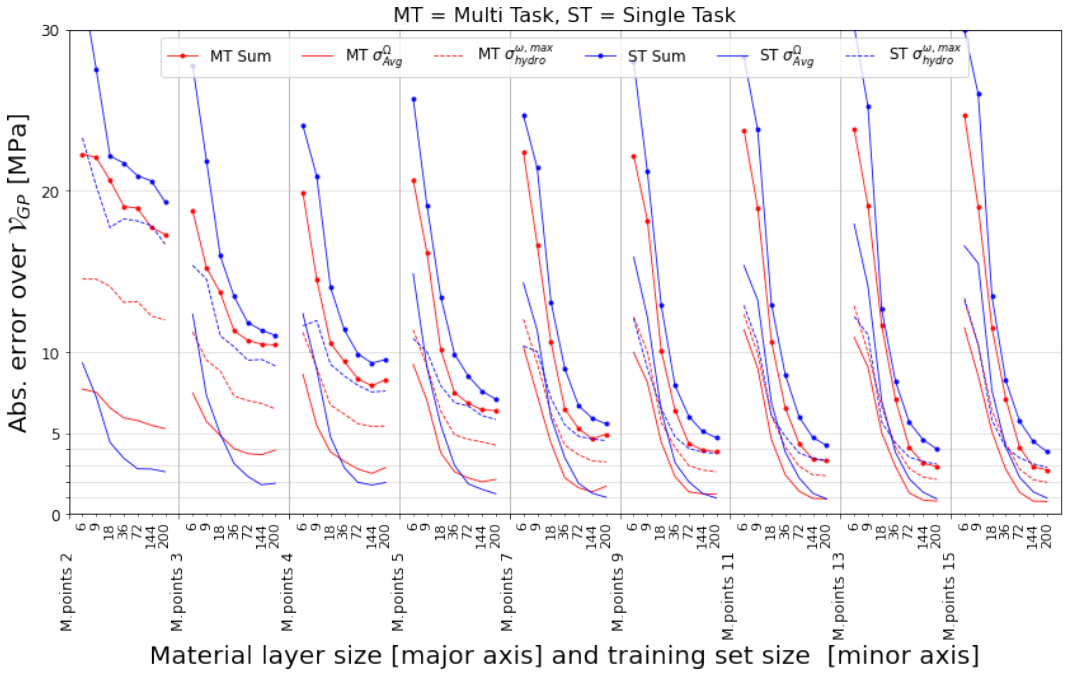


Figure 6.5: Comparison of the single- and multitask losses for GP data: average error over 10 networks

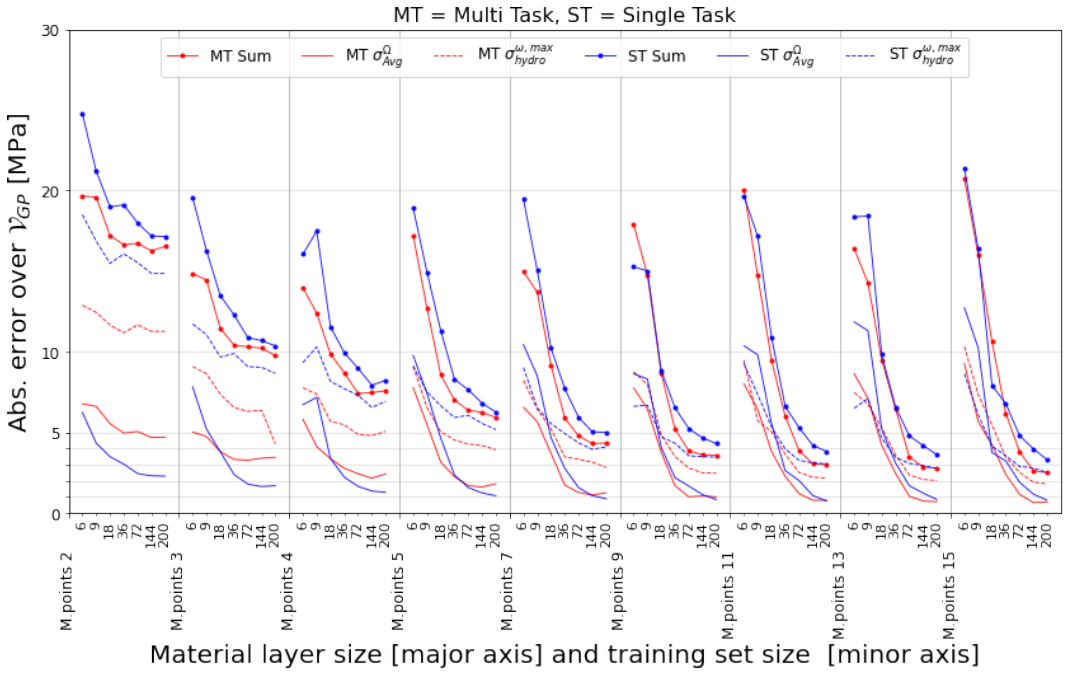


Figure 6.6: Comparison of the single- and multitask losses for GP data: minimum error over 10 networks

The overall conclusion that can be drawn is that the multi-task model performs better then the 2 single-task models as it finds a better balance between the losses for the average and the maximum stress.

This feature is most notable for the models with a low number of material points: the two red lines (losses from the multi-task model) are between the blue lines (losses from the single task models). The single-task model predicting average stresses performs very well for the case with only 2 material points. The single-task model predicting the maximum stress needs a couple more points to start predicting well. As soon as the number of material points is increased the models start to converge. The average error of the single-task and multi-task converge quite well, while the error for the maximum stress remains predicted better by the multi-task model.

The multi-task model predicts the maximum stresses better than the single-task model predicting the maximum stresses, which is most evident for the smaller networks. One reason is because the multi-task network also has to make predictions on the average stress. It is very likely that stresses predicted by one (or more) material point(s) is larger than the average stress and therefore the error for maximum stresses is somewhat limited. This statement becomes more clear in the next section, where the distribution of stresses in the material layer is investigated.

As it seems like the multi-task model finds a balance between the maximum and the average stress, the distribution of the stresses predicted by the multi-task model is the next point of interest. The next section will look into the distribution of stresses in the material layer and see how the material layer is able to accurately predict both stress quantities.

6.3. Distribution of stresses

The full-field microscopic stresses are computed for each sample for 60 timesteps. The distribution of the microscopic stresses at any timestep can be displayed by creating a histogram and deriving the probability density function (PDF). First, all occurring stresses are collected in bins with a certain bin range. The larger the number of bins, the more precise the distribution. The number of bins is set to 25, which in most cases result in a bin width ranging from 1-3 MPa. The bins are shown in blue in Figure 6.7. The height of the bin represents how often the stress of the given stress range occurs. Histograms give a rough sense of the density of the underlying distribution of the data. This distribution is plotted as a blue continuous line. The minimum and maximum of the full-field stresses are found (min($\mathcal{D}$), max($\mathcal{D}$)) and are shown in green. These lines indicate the limits of the data for each timestep.

The stress distribution is then to be compared to the predicted hydrostatic stresses from the PRNN presented in Section 6.1. To do so, the hydrostatic stresses of all fictitious material points are calculated. The data used in Figure 6.7 is from a model with 11 material points. For each timestep the material layer computes 33 stress components which result in 11 hydrostatic stresses. These predictions are presented in Figure 6.7 as a histogram with 11 red bars.

Figure 6.7 shows the full-field stresses and the hydrostatic stresses that are computed from the fictitious stresses in the material layer for 4 randomly selected timesteps. The first plot is after the first strain increment at t = 1. The properties of the model have the assumption that the material is at rest and there are no residual stresses. The histogram of plot 1 shows this assumption is satisfied as the stresses are centered around 0 MPa. A few timesteps later, the stress in the material is gradually building up before the stresses turn mostly negative in the final plots.

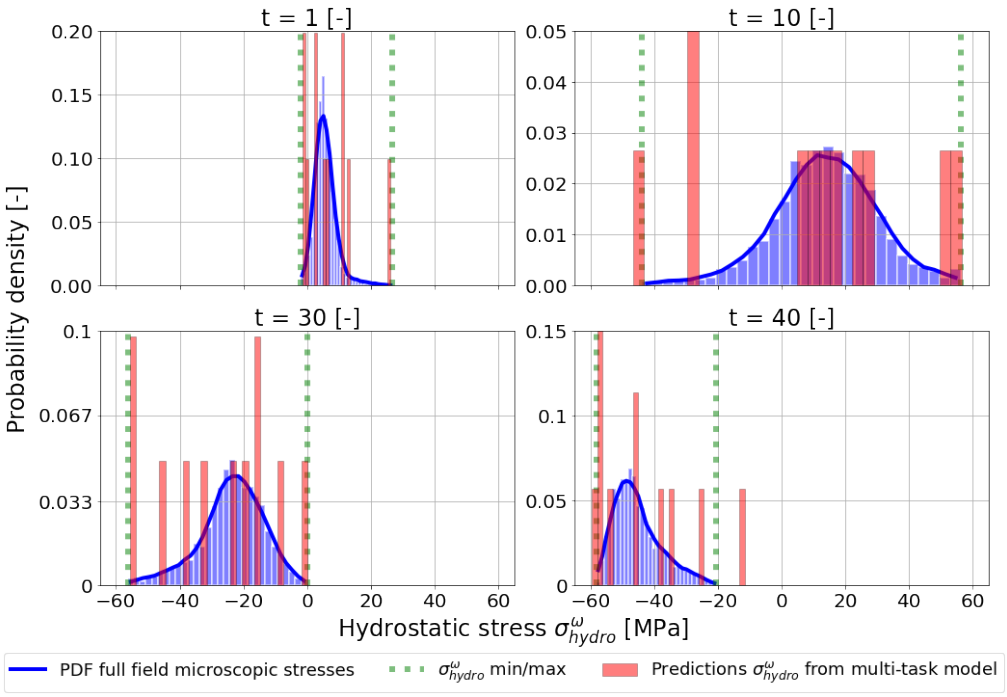


Figure 6.7: The full-field microscopic hydrostatic stresses versus the predicted stresses from the multi-task model

It is hard to make statistically sound statements about the distribution of the predicted stresses as there are only 11 datapoints per timestep. However, when analysing the 4 plots, it seems the bulk of the predicted stresses match the stress values where there is a larger probability density for the full-field stresses. Furthermore, it is remarkable how the limits of the predictions of the hydrostatic stresses follow the limits of the full-field stresses. Based on the results in Section 6.1 it already became clear that the maximum hydrostatic stresses can be predicted accurately. The visual representation of the stress distribution versus the predictions gives additional insights: the hydrostatic stress minimum matches the minimum of the occurring full-field stress. This suggests that predicting the minimum stress can be an additional quantity that the current network is able to predict. A proof of concept of this multi-task model predicting the average stresses and both limits of the hydrostatic stresses is shown in Chapter 7.

6.4. Cross validation

The advantages of performing a cross validation are discussed in Section 4.4. In general, the same conclusions can be drawn from the cross validation of the multi-task approach:

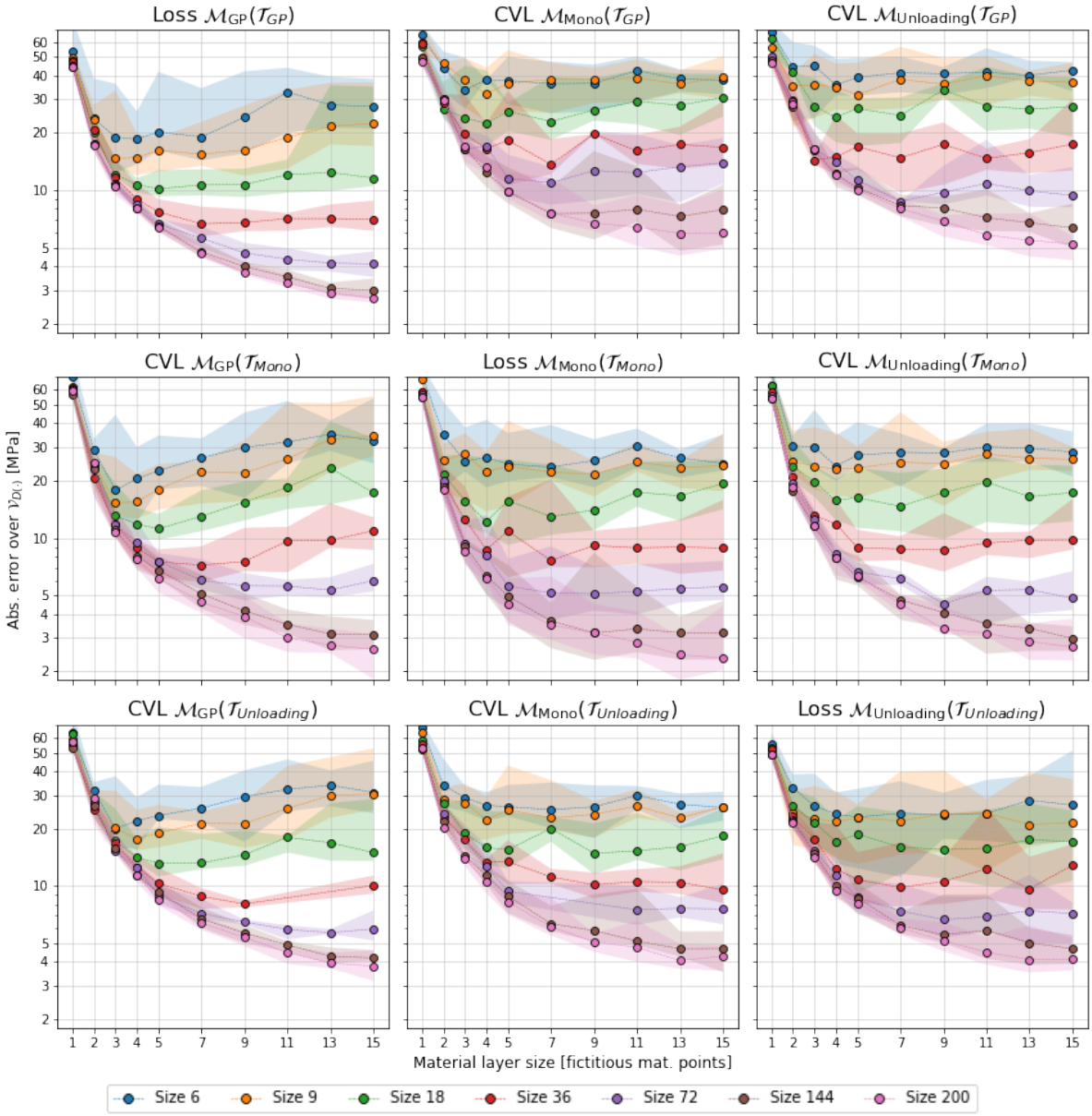


Figure 6.8: Cross validation for the multi-task models

- The models trained on monotonic and unloading data show a worse performance on the test set with GP data as these models fail to generalize to characteristics of the GP dataset. The errors for models with equal training size and material layer size are about double of monotonic/unloading compared to a models trained on GP data.
- When looking at row 2 and 3, the performance of all three models look quite comparable. In most cases the model trained on GP data shows a more narrow range of errors per combination. This suggests a better training stability and model robustness.
- In comparison to the cross validation of Section 4.4 the absolute error over the validation dataset increases when adding more fictitious material points. In the previous comparison the error only showed a stagnating error. This characteristic comes from the addition of the average loss to the

loss function. The PRNN by Maia et al. and the PRNN with cohesive material points by Kovács showed that any additional material points above a certain number made the network more prone to overfitting. This characteristic is more evident for the models with a limited amount of training curves. In case more training curves are added, the training data is more diverse and the model shows a better response. In case the number of material points is increased, the expectation is that the models with a high amount of training data will over fit as well.

- Overall, the multi-task model shows to perform very well. For some cases, the loss is less than 2 MPa. This is the loss of the average and the maximum stress combined.
- There was no noticeable difference in computational time of the multi-task approach compared to single task models.

6.5. Stress-strain curves

The previous sections showed that the multi-task approach showed good performance and has the potential to include more quantities like the minimum stress. In this section one model will be selected to plot stress-strain curves and see what predictions the network is able to make.

Just as in Section 4.5, 11 fictitious material points perform well for both the GP dataset and the monotonic dataset. The learning curves for this number of fictitious material points is shown in Figure 6.9. This analysis will give more insight in how many training curves are needed to converge. For this analysis, 72 training curves seems appropriate. For both the GP and unloading dataset this number is plenty to reach convergence. The model trained on monotonic data does still improve with more curves, but 72 curves is a good balance in computational time and performance.

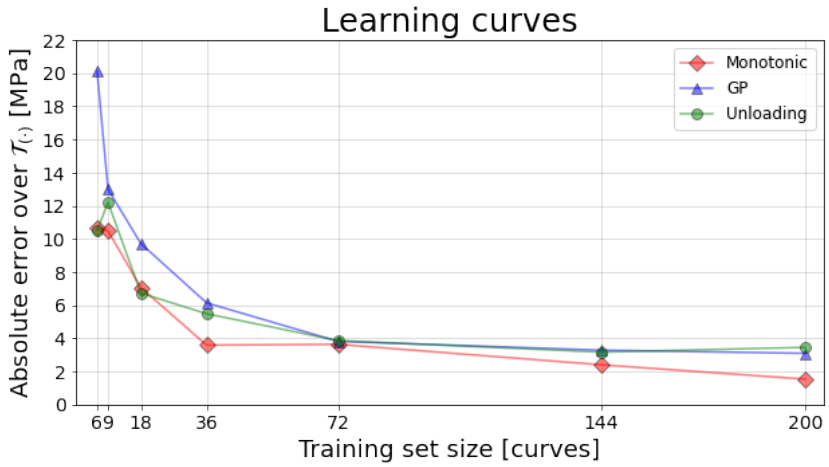


Figure 6.9: Learning curve for the best multi-task models with 11 material points and different training loading types

The selected model has 11 fictitious material points and used 72 training curves. A test set of 100 samples is then used to make predictions. The cross validation from Section 6.4 showed that the model trained with GP performs equally or better when predicting the stress of a monotonic and unloading sample than the models that are trained on monotonic and unloading data. This is the reason the model trained on GP data is used to predict samples from GP, monotonic and unloading samples. From 100 samples of each dataset a representative stress strain curve is selected. A representative sample is one that lies within the IQR of the error distribution and preferably close to the median. The distribution of predicted stresses for each sample is shown in Figure 6.10 and the sample that is selected is marked with a star. The stress strain curves are shown in Figure 6.11 for the 3 different samples of each dataset.

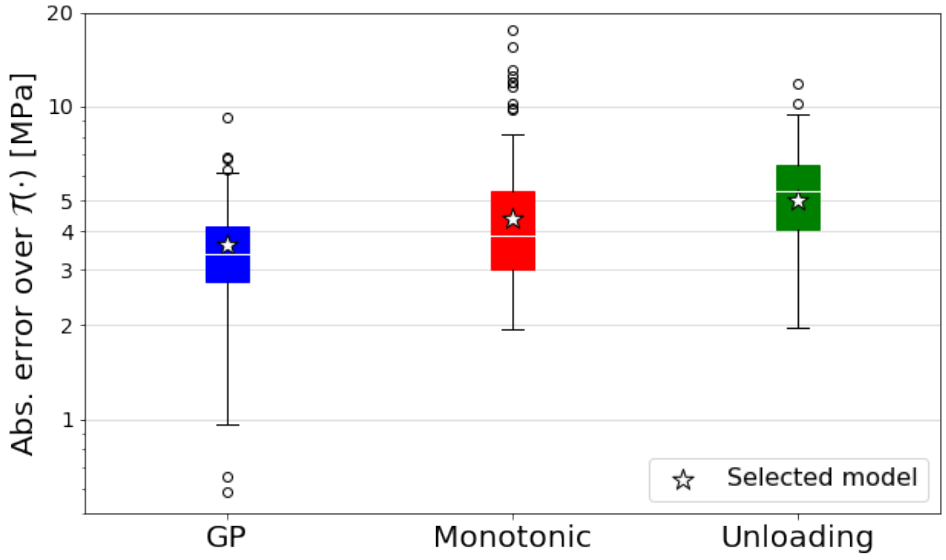


Figure 6.10: Distribution of errors over test set with 100 loading curves of different types evaluated by multi-task PRNN with 11 material points and trained on 72 training curves

The black data points in Figure 6.11 indicate the target of the model and the red data points show the predictions made by the model. The first 3 plots are the stress strain curves in x, y and xy-direction. Together these form the average stress quantity. The last plot shows the prediction made on the hydrostatic stress over the macro hydrostatic strain.

The first set of plots is the prediction of a GP sample, the second set of plots shows the monotonic sample and the last set of plots is the prediction of the unloading sample. Overall, the predictions follow the shape of the stress strain paths, with a couple of notes on the plots:

- It is hard to explain the results of the GP dataset because the magnitude and step size is random for each time step. The stress strain paths show random directions and each sample is different. The predictions follow the target closely and the results are very promising. The overall loss of this sample is the combined loss of the average stress and the maximum hydrostatic stress, which is only 3.61 MPa for this sample. Reminding this sample is a representative sample and the predictions can get even closer to the target.
- The predictions of the monotonic sample shows interesting behavior. The monotonic dataset for all samples show an increasing (or purely decreasing) stress value until it reaches a limit. This characteristic can be seen when looking at the target (the black data points). The predictions of the GP model do not reflect this characteristic as the stress shows a clear maximum before decreasing, visible in the plots in x and xy-direction. This is also visible in the final set of plots, where the GP model made predictions on a random unloading sample.

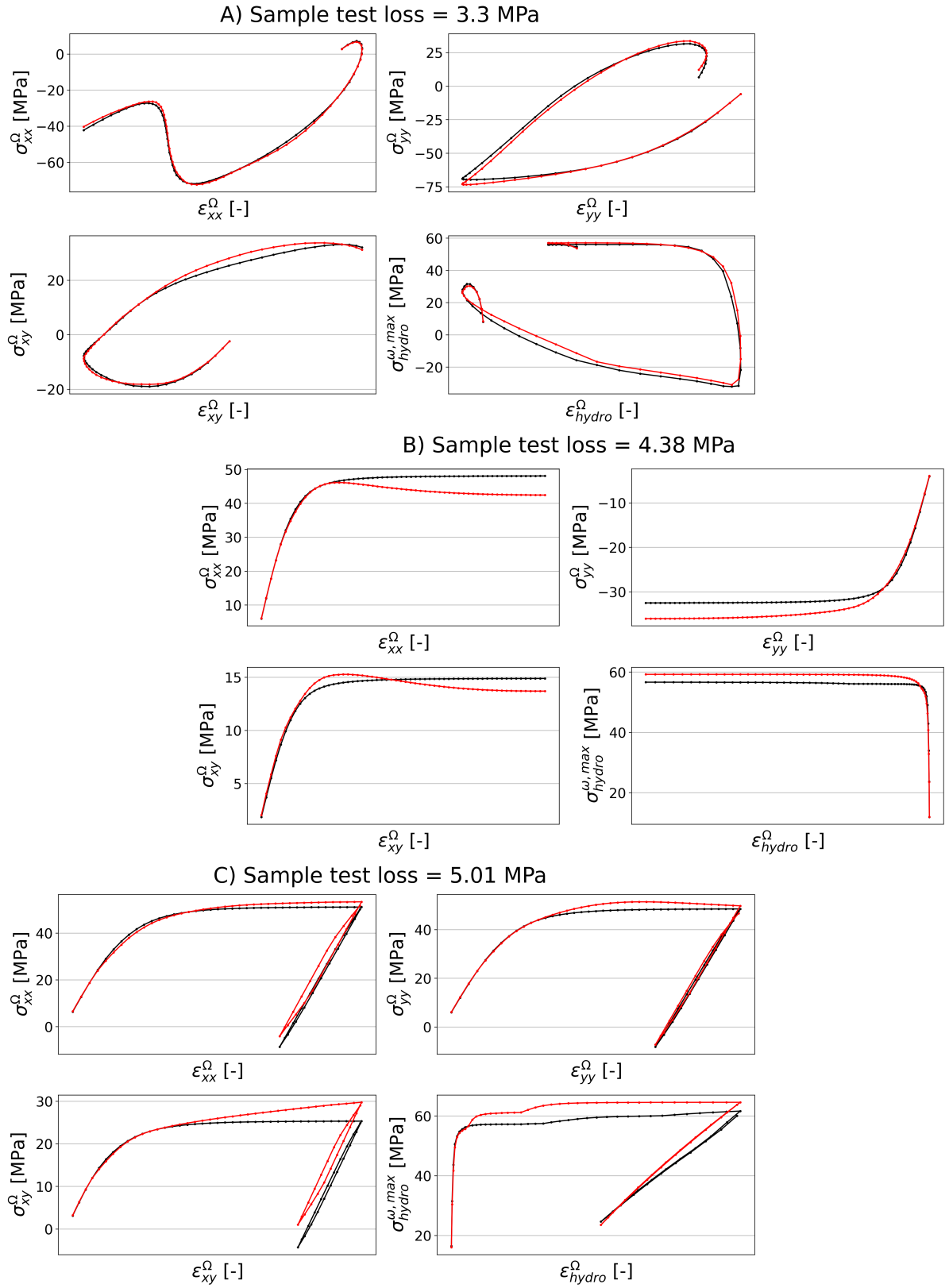


Figure 6.11: Stress strain paths for a representative GP, monotonic and random unloading sample

7 | Triple-task model

The multi-task approach showed that the network can compute two quantities in parallel and compute a combined loss. There are no additional trainable parameters, so the increase in computational time to train models is limited. Section 6.3 showed that the limits of the predictions of the multi-task model matched the limits of the full-field stresses well. This resulted in the idea that the minimum hydrostatic stress can be added to the model in the same way the maximum stress was added. The results are presented in this chapter.

7.1. Architecture

The proposed architecture is shown in Figure 7.1. It includes the same encoder and material layer as seen in earlier chapters and the decoder from the PRNN in[14] to predict the first set of quantities in the loss function. The hydrostatic stress is computed in the material layer and the maximum and minimum values are found. These values are added to the loss function. The loss function is now an addition of 3 individual losses and no scaling is applied as the individual losses have the same order of magnitude. In future work more complex loss functions can be researched. The solid lines in Figure 7.1 indicate there is no training done, whereas the dashed lines indicate there are weights attached between the nodes.

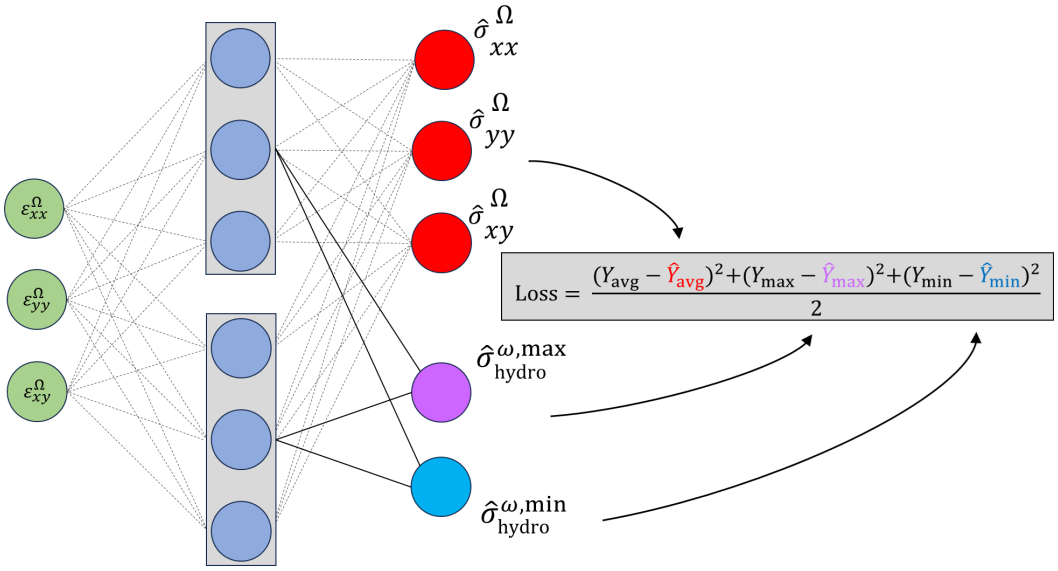


Figure 7.1: The architecture for the multi-task model predicting 3 the average stress and the hydrostatic limits

It is a deliberate choice to compute both the minimum and the maximum hydrostatic stress. One could argue to take the absolute value of all stresses and compute the maximum because this will reduce the computational time and complexity of the loss function. However, by taking the absolute value a lot of information of the stress state is lost. Most material properties have different characteristics when in tension or compression, so having the additional information if the material is in compression or tension is useful.

The datasets used so far lack the governing hydrostatic minimum stresses. The GP dataset is the most general dataset that is used in this work and the networks trained so far showed the best results when trained on the GP dataset. For this final analysis, the GP dataset is extended with a column presenting the minimum hydrostatic stress for all timesteps and samples.

7.2. Results

The result of the model selection procedure is shown in Figure 7.2. The reasoning for the number of fictitious material points and the number of training curves is explained thoroughly in Section 4.1 and the same values are used here. Other model parameters like the numbers of epochs and the batch size remain unchanged.

The results show differences compared to the other results on GP data for the 2-task model. This time the range of errors are larger and overlap other results. This makes it harder to identify how this new architecture performs. When looking at the average of each combination of training size and material layer size, most of data is found ranging from 10 - 30 MPa. It also appears that the error does not drop as quickly when increasing the number of training curves in comparison to previous models. Furthermore, the range of errors seems higher.

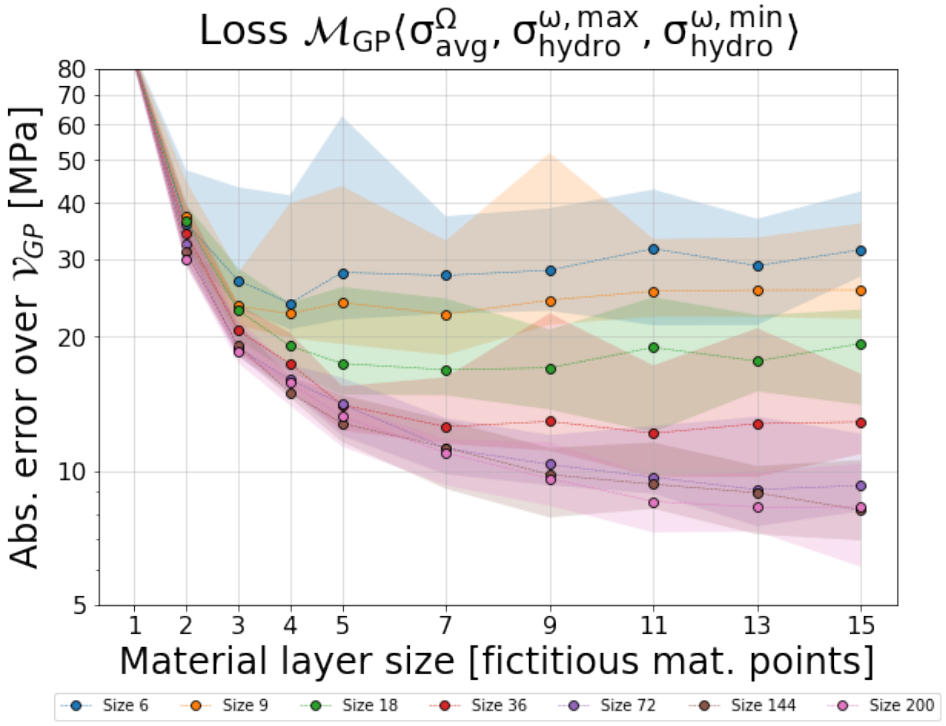


Figure 7.2: The validation error of the triple-task model trained on GP data

Reminding that the error that is showed is the combined error over 3 individual losses, some models show a loss of just 7 MPa which is very low. Figure 7.3 and Figure 7.4 show the exact same data as in Figure 7.2 but here the losses are split up in 3 parts. The first figure shows the effect of the material layer size on the major x-axis. Every gridline indicates an increase of material layer size. Each bar consists of 3 colors that indicate the average error, the maximum hydrostatic error and the minimum hydrostatic error. All the bars between the vertical gridlines (the minor x-axis) show the increase in training size. Figure 7.4 shows the same data but with the major and minor x-axes reversed. These plots help to visualize the data more effectively as the model selection results overlapped in Figure 7.2.

The results show the average errors over 10 initiations of the same network, trained on a different training set. In general the conclusion can be drawn that the larger the layer size and or training set size, the lower the error. However, the loss on the average stress is substantially lower than the other two errors. The error of the maximum and minimum hydrostatic stress does improve when the layer size and or training set size is larger but the losses do stagnate.

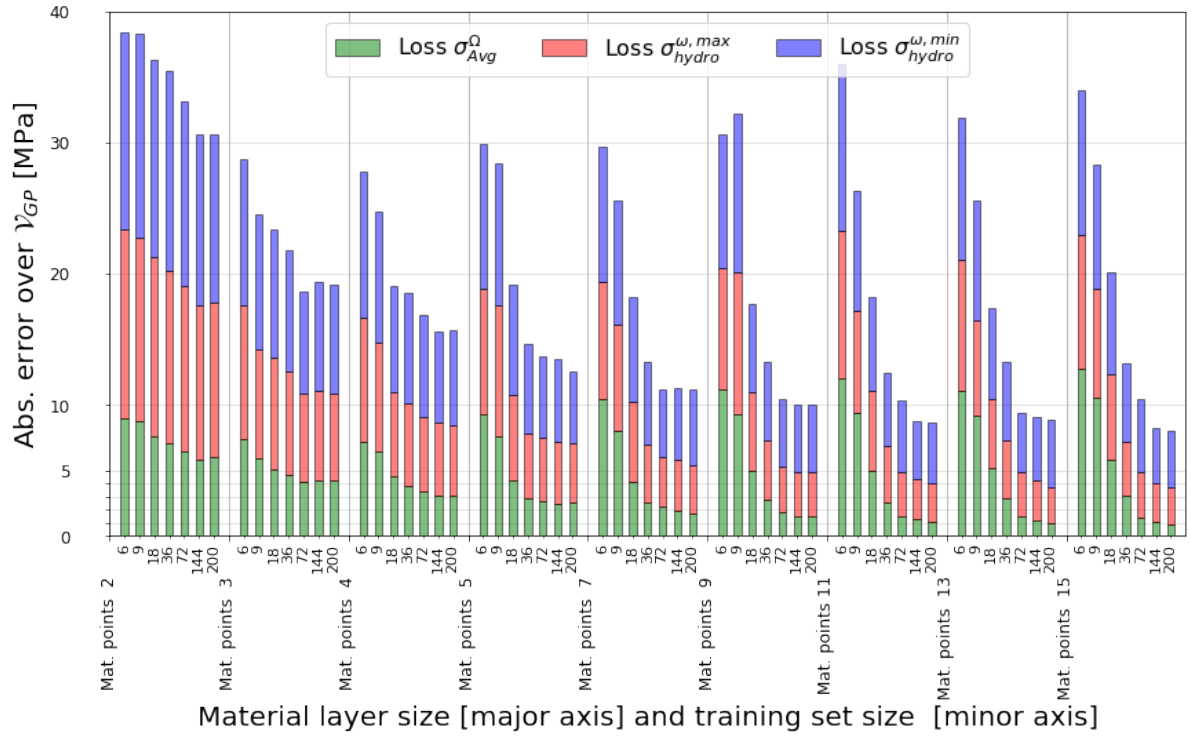


Figure 7.3: Model selection of the triple-task model: material layer size versus training set size

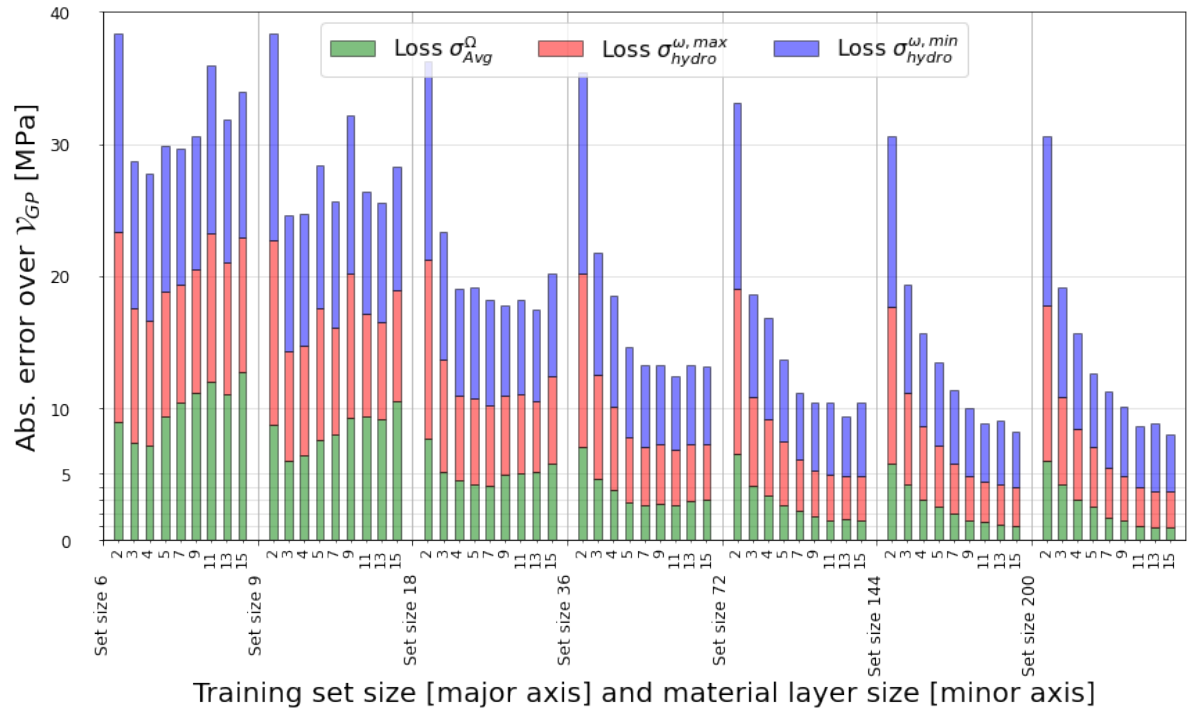


Figure 7.4: Model selection of the triple-task model: training set size versus material layer size

The following figures show the comparison between the individual losses of the triple-task model versus the losses of the 3 single-task models. Figure 7.5 shows the average over 10 initiations of each combination of layer size and training set size and Figure 7.6 shows the best performing models (the models with the lowest error). When looking at the first figure, it becomes clear that the triple-task model outperforms the 3 individual models because the single-task predicting the minimum hydrostatic stress performs quite poorly. The multi-task model is much better able to predict the minimum hydrostatic stress as it seems to find a good balance in predicting the three stresses. Section 6.3 discussed the distribution of the prediction versus the distribution of the full field stresses. Here it became evident that by combining the 2 tasks in a single model, the limits of the full field stresses matched with the limits of the predictions. This causes the triple-task model to predict the minimum hydrostatic stress much better than the single task model does.

The single task models predicting the average and maximum stress perform slightly better than the triple-task model for a lower number of material points. For a larger number of points, the results converge and there is no difference in the single task as the triple-task approach.

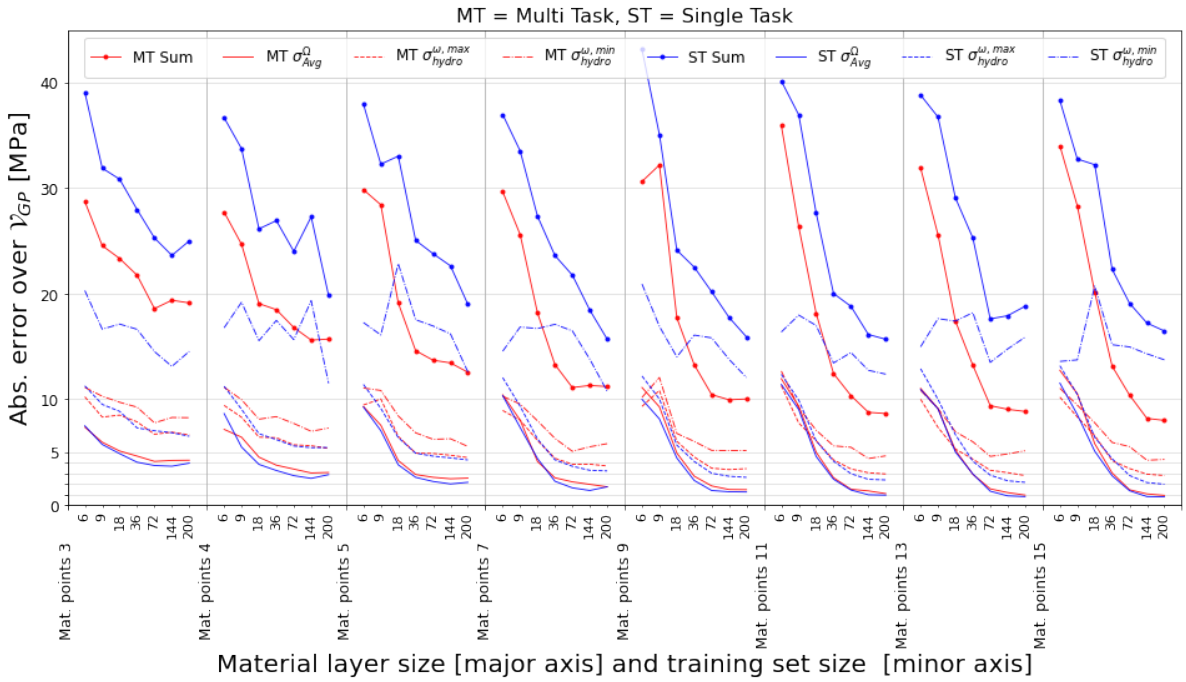


Figure 7.5: Comparison of individual losses for triple-task and single-task models: the average error over 10 networks

Figure 7.6 shows the best performing models instead of the average over 10 models. When looking at the results for the minimum hydrostatic stress, there is no clear trend visible for the single task model (the blue dash-dotted line). This model seems to show better performance when the number of training curves used increase, but this behavior is not visible for a larger amount of material points in the network. The triple-task model shows a more clear trend in predicting the minimum stress, which is in line with the other predictions.

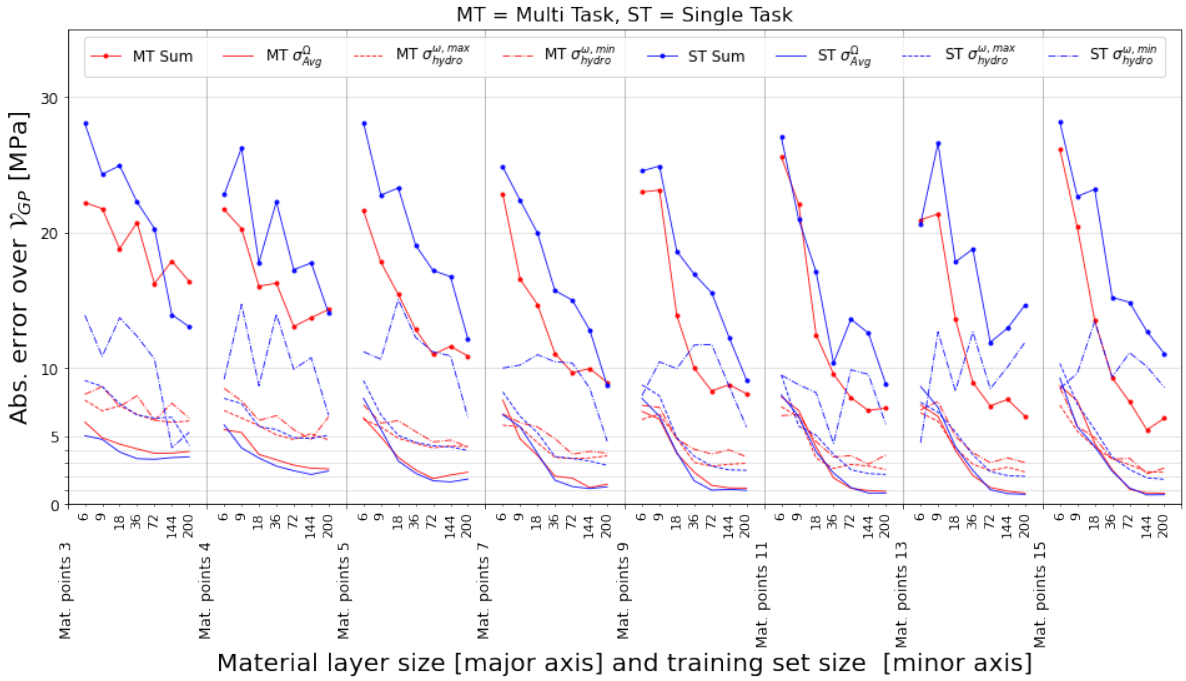


Figure 7.6: Comparison of individual losses for triple-task and single-task models: the minimum error over 10 networks

When looking at the results of the single-task model predicting the minimum hydrostatic stress, it shows that the model struggles to converge to a low error, even when more points or training curves are added. The full model selection is therefore shown in Figure 7.7 to see how the architecture performs. Unfortunately, this model does not predict as well as the other single-task models. The error is stuck in the range between 10-15 MPa. Predicting the minimum hydrostatic microscopic stresses is not in the scope of this thesis and therefore no further optimization is done on this network. However, it remains impressive that the multi-task model is able to predict the minimum stresses accurately as it finds a good balance in predicting the three individual stress quantities.

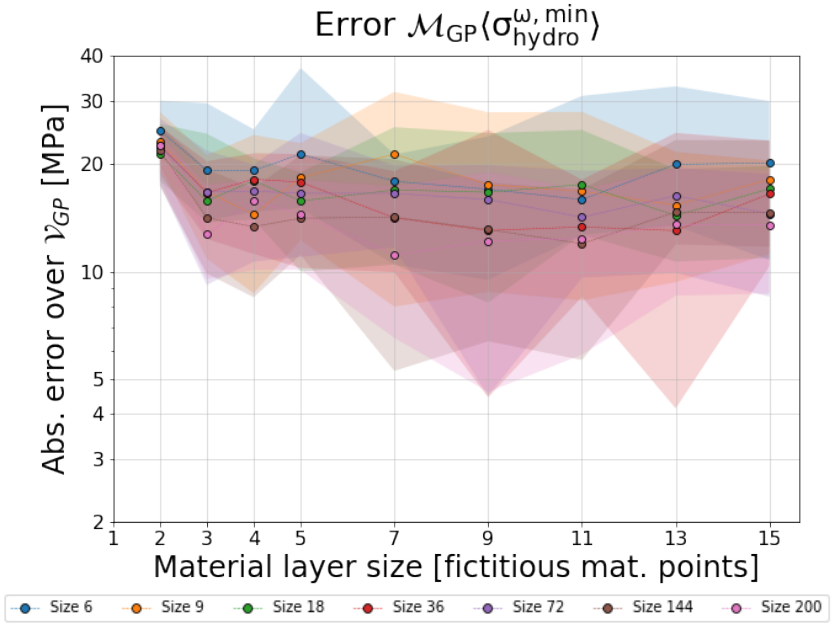


Figure 7.7: Model selection of the single-task model predicting the minimum hydrostatic stress

Based on the results from Figure 7.3 and 7.4 a model is selected with 9 material points and 36 curves used during training. 100 GP strain paths are sampled and form a test set. This test set is used to make predictions. Figure 7.8 shows the distribution of the error and a representative sample is selected that is near the median. All stress-strain curves are plotted for this representative sample. The first 3 plots are the curves in x, y and xy-direction. The bottom 2 plots show the predictions versus the target for the maximum and minimum hydrostatic stress.

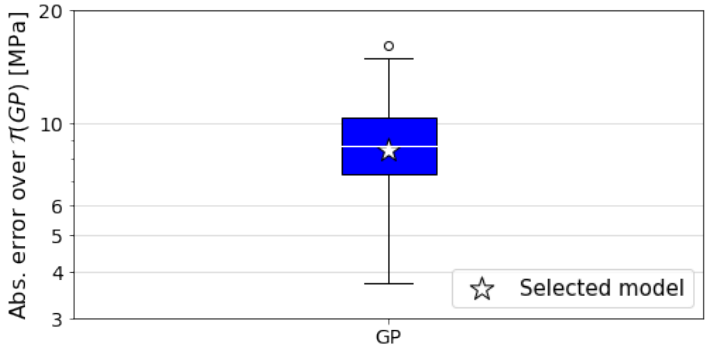


Figure 7.8: Distribution of stresses of 100 GP test samples

The stress-strain curves are shown in Figure 7.9. The presented sample has a total error of 8.46 MPa which consists of 2.45, 3.33 and 2.67 MPa for the average, maximum and minimum error respectively. In this case the individual errors are comparable. In other samples in this test set the distribution is less evenly spread. The loss function values the individual losses evenly but a weighted loss function can be applied to emphasize for example the maximum loss.

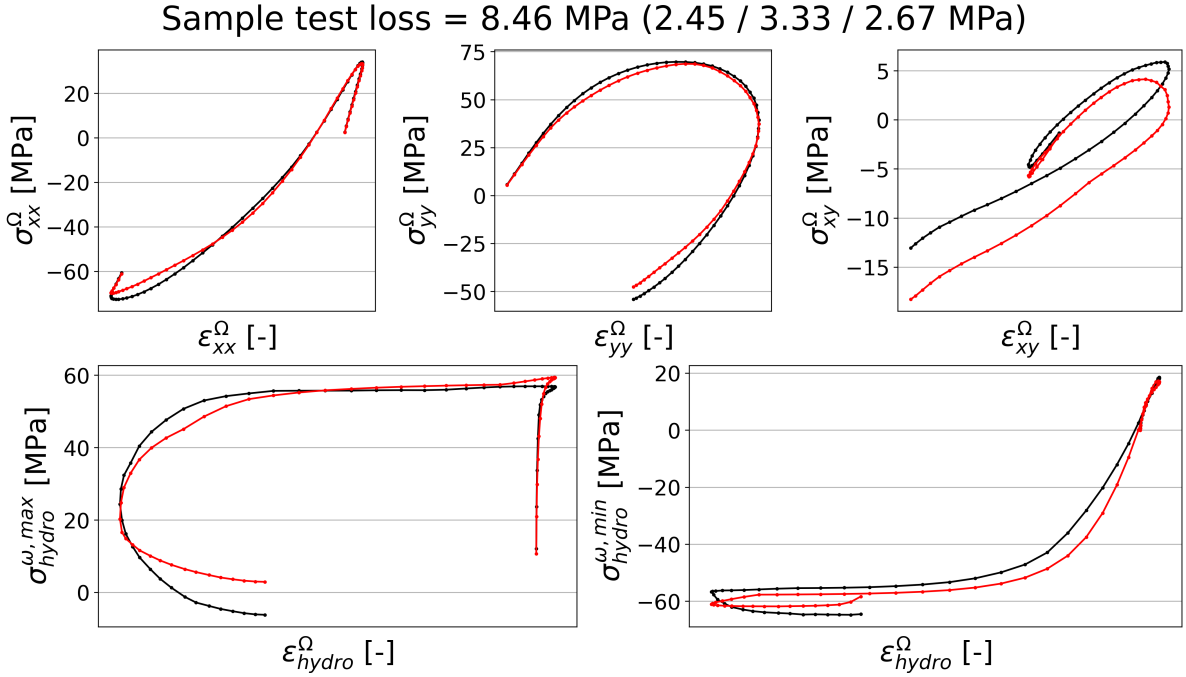


Figure 7.9: The prediction of a GP sample with a combined loss of 8.61 MPa

This marks the end of this chapter. Once again the model has proven to perform well with a limited amount of material points, training curves and epochs. The analysis done in this chapter was only performed on the GP dataset and there was no cross-validation done here as the dataset was not extended with the minimal error for the monotonic and unloading dataset. It is expected that these show the same results as for earlier models.

8 | Conclusion and recommendations

The main objective of this research is to design a surrogate model that can accurately predict the local maximum stress of a composite material at microscale and to leverage the knowledge of existing models into new models. The final chapter of this research provides an answer to all the research questions and formulate the findings on the main research question. Recommendations for future research are provided in the last section.

8.1. Conclusion

In order to conclude this research, the main research question *"To what extent is a modified PRNN able to accurately predict local maximum stress values in heterogeneous microscopic material models?"* is to be answered. The sub questions are answered first which will lead to the final conclusion.

How to efficiently incorporate the maximum stress in the network?
In total three different architectures have been designed in Chapter 4 to include predictions on the maximum hydrostatic stress. The first two architectures contain a decoder, but it became evident that having a decoder in the network averages the stresses from all material points. When selecting the maximum stress from the RVE, just one point is responsible for the maximum stress. Therefore the third architecture presented in this thesis replaces the decoder from the previous architectures by a function (without trainable parameters) that simply selects the maximum stress from the material points, which is an analogy to finding the maximum stress in the RVE. The amount of trainable parameters in this model is very low. The 3 input nodes are full connected to 3 nodes in m number of material points. The model selection shows that just 5 points are needed in case of training with a low amount of curves, resulting in only 45 trainable parameters. When more training curves are used during training, the model selection shows that $7-9$ points are optimal and the error is reduced by roughly 30%. The cross-validation shows this network can accurately predict on unseen loading scenarios which results in this model being robust.

In what ways can transfer learning be applied to leverage the knowledge of existing models in creating new models?
Chapter 5 shows how transfer learning is applied to leverage knowledge from a model trained on task 1 to task 2. This process is reversed as well. The results for both directions are quite similar in terms of error for direct training versus transfer learning. In the case of transfer learning from a model trained on prediction maximum stresses to the average stress, transfer learning with freezing the encoder delivers surprising results. The hypothesis was that this model is too rigid to perform well for the second task with a frozen encoder, but results show that this model performs at mostly equal or better than warm-starting or directly training as can be seen in Figure 5.3. The encoder works very well for both tasks and freezing the encoder leads to only training the decoder. This results in the model fine tuning the decoder in such way that the model is able to perform better. However, results between direct training and transfer learning are close and no clear favorite approach can be pointed out regarding the final error.

Transfer learning does however speed up training as convergence is reached faster. As both direct training and transfer learning are fairly small networks, training is quick for either framework resulting in a limited computational benefit when applying transfer learning.

One identified challenge was that the optimal number of material points for task 1 differs from the ideal number of material points for task 2. The analysis in Section 5.4 shows that this is solved by sampling new weights from the encoder of model 1 to create an encoder with appropriate size for model 2. The results indicate that this sampled encoder performs equal or slightly better in comparison to a

sub-optimal material layer size for task 1.

A shortcoming from transfer learning is identified in Section 5.3 where a model is trained on a different sampled training set, resulting in 10 different encoders. When these encoders are transferred from model 1 to model 2, the result is that the best performing encoder in model 1 does not guarantee the best result for model 2. Selecting the right encoder for model 2 is therefore tricky to execute.

Can the current architecture be leveraged to a multi-task approach?
Chapters 4 and 5 show that the original PRNN and the network in Section 4.3 use the same encoder and material layer. Chapter 6 explores a framework with a multi-task approach to combine the tasks in a single model. The model selection shows excellent results: the multi-task model outperforms the 2 individual models in most cases. This is because the model finds a good balance in predicting average and maximum stresses together. The single-task model predicting the average stress performs better for a layer size smaller than 7, but the results for average stresses converge for larger material layers. Reminding that the number of trainable parameters for the multi-task approach is just two thirds of the number of trainable weights for the 2 single-task models combined, this result is even more impressive.

Another interesting finding is in the distribution of the stresses in the material models plotted versus the full field stress distribution in Section 6.3. The limits of the predictions match the limits of the full-field microscopic stresses closely. This finding lead to the idea on adding a third task to the model: predicting the minimum local stress as well. The results are presented in Chapter 7 and show the same trend as for the multi-task approach. The triple-task model outperforms the 3 single-task models as it is able to find a better balance in predicting the 3 stress quantities. It seems that by combining the tasks, the model is able to accurately finds a balance between all stress quantities which results in a lower error. Especially the single-task model predicting minimum stresses seems to be less robust as the error does not converge as smoothly as the other 2 models. The model selection of this single-task architecture, presented in Figure 7.7, shows that the predicted stress is actually stuck in the range of 10-15 MPa. In case this single-task model is optimized, the three single-task models and the triple-task model should converge. In the end, the triple-task model has only half the amount of trainable parameters as the original PRNN and even less than the three single-task models combined, which is impressive.

This leads to the conclusion of the main research question. The PRNN that selects the maximum stress from the material layer without decoder (Section 4.3) shows an outstanding performance. Apart from the complex physical relations in the material layer, the architecture is very basic with a very limited amount of trainable parameters. This results in low computational power and fast training. Furthermore, the model selection shows that the model performs well with a limited amount of training curves. Finally, the cross validation shows that the model is both robust and flexible to adapt to unseen loading scenarios.

8.2. Recommendations

In this research, it has been shown that the model predicting average macro stresses can be leveraged to also extract the maximum and minimum hydrostatic stresses at microscale. The model currently only states the maximum or minimum stresses. One recommendation is to further implement the local stress limits to compute material failure. This can be done by adding more physical laws in the model, just as is done in the material layer.

A second recommendation is to explore if this architecture can be expanded to 3D models. The RVE needs to be expanded to 3D and additional boundary conditions have to be formulated. The network needs expansion as well: 6 inputs must be defined because there are 6 strain directions in 3D. Each material point must also be expanded to output 6 stress outputs.

The multi-task approach in this thesis showed that the PRNN can be used as a modular network in which more material properties can be assigned and learned. Future research can explore if other quantities from the full microscopic field can be learned as more data about the stress distribution in

the material layer (e.g. average plastic strain deformation) is fed into the network.

The single-task model predicting the minimum hydrostatic microscopic stresses did not converge as well as the single-task model predicting the maximum hydrostatic stresses. Future research can investigate why this model does not perform as well and possibly optimize this model further. The comparison between the triple-task model and this optimized single-task model will be a better comparison than done in this thesis.

Another topic of research is to investigate if the distribution of the fictitious stresses can match the stresses of the RVE. Section 6.3 shows that the limits of the fictitious stresses follow the stresses of the RVE quite well but there are too little material points to create a distribution.

Finally, this research did not manage to use the gained knowledge of the new models on a civil engineering use case. An active field of research in the civil engineering domain is using surrogate models for real-time monitoring and predictive maintenance. Future research could focus on expanding the models developed in this research that are robust enough for continuous, real-time data input, enabling them to predict stress and failure points under live loading conditions and facilitate preventive maintenance of composite structures. Hopefully more practical examples can be used to inspire other students, engineers and researchers to explore the options of using machine learning in the civil engineering domain even more!

Bibliography

[1] IEA. Net zero by 2050, iea, paris, 2022.

[2] Jouni Korhonen, Antero Honkasalo, and Jyri Seppälä. Circular economy: The concept and its limitations. *Ecological Economics*, 143:37–46, 2018. ISSN 0921-8009. doi: https://doi.org/10.1016/j.ecolecon.2017.06.041. URL https://www.sciencedirect.com/science/article/pii/S0921800916300325.

[3] Roderick Campbell. China in the early bronze age: Shang civilization by robert l. thorp (review). *Asian Perspectives*, 51:313–321, 01 2014. doi: 10.1353/asi.2014.0007.

[4] Sharun Hegde, B. Satish Shenoy, and K.N. Chethan. Review on carbon fiber reinforced polymer (cfrp) and their mechanical performance. *Materials Today: Proceedings*, 19:658–662, 2019. ISSN 2214-7853. doi: https://doi.org/10.1016/j.matpr.2019.07.749. URL https://www.sciencedirect.com/science/article/pii/S2214785319330445. 1st International Conference on Manufacturing, Material Science and Engineering.

[5] J. Tinsley Oden, Kumar Vemaganti, and Nicolas Moës. Hierarchical modeling of heterogeneous solids. *Computer Methods in Applied Mechanics and Engineering*, 172(1):3–25, 1999. ISSN 0045-7825. doi: https://doi.org/10.1016/S0045-7825(98)00224-2. URL https://www.sciencedirect.com/science/article/pii/S0045782598002242.

[6] Karthikayen Raju, Tong-Earn Tay, and Vincent Beng Chye Tan. A review of the fe2 method for composites. *Multiscale and Multidisciplinary Modeling, Experiments and Design*, 4(1):1–24, Mar 2021. ISSN 2520-8179. doi: 10.1007/s41939-020-00087-x. URL https://doi.org/10.1007/s41939-020-00087-x.

[7] Frédéric Feyel. Multiscale fe2 elastoviscoplastic analysis of composite structures. *Computational Materials Science*, 16(1):344–354, 1999. ISSN 0927-0256. doi: https://doi.org/10.1016/S0927-0256(99)00077-4. URL https://www.sciencedirect.com/science/article/pii/S0927025699000774.

[8] M.G.D. Geers, V.G. Kouznetsova, and W.A.M. Brekelmans. Multi-scale computational homogenization: Trends and challenges. *Journal of Computational and Applied Mathematics*, 234(7):2175–2182, 2010. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2009.08.077. URL https://www.sciencedirect.com/science/article/pii/S0377042709005536. Fourth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2008).

[9] Shaoheng Guan, Xue Zhang, Sascha Ranftl, and Tongming Qu. A neural network-based material cell for elastoplasticity and its performance in fe analyses of boundary value problems. *International Journal of Plasticity*, 171:103811, 2023. ISSN 0749-6419. doi: https://doi.org/10.1016/j.ijplas.2023.103811. URL https://www.sciencedirect.com/science/article/pii/S0749641923002954.

[10] Hernan J. Logarzo, German Capuano, and Julian J. Rimoli. Smart constitutive laws: Inelastic homogenization through machine learning. *Computer Methods in Applied Mechanics and Engineering*, 373:113482, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113482. URL https://www.sciencedirect.com/science/article/pii/S0045782520306678.

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

[12] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[13] Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2021.113741. URL https://www.sciencedirect.com/science/article/pii/S0045782521000773.

[14] M.A. Maia, I.B.C.M. Rocha, P. Kerfriden, and F.P. van der Meer. Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate. *Computer Methods in Applied Mechanics and Engineering*, 407:115934, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2023.115934. URL https://www.sciencedirect.com/science/article/pii/S0045782523000579.

[15] H.L. Cheung and M. Mirkhalaf. A multi-fidelity data-driven model for highly accurate and computationally efficient modeling of short fiber composites. *Composites Science and Technology*, 246, 2024. doi: https://doi.org/10.1016/j.compscitech.2023.110359. URL (https://www.sciencedirect.com/science/article/pii/S0266353823004530).

[16] Y Ban, J Hou, X Wang, and G Zhao. An effective multitask neural networks for predicting mechanical properties of steel. *Materials Letters*, 353, 2023. ISSN 0167-577X. URL https://doi.org/10.1016/j.matlet.2023.135236.

[17] T Iraki, L Morand, J Dornheim, N Link, and D Helm. A multi-task learning-based optimization approach for finding diverse sets of material microstructures with desired properties and its application to texture optimization, 2022. URL https://arxiv.org/abs/2111.00916.

[18] Coen Clarijs. *Mechanical performance of glassy polymers : influence of physical ageing and molecular architecture*. Phd thesis 1 (research tu/e / graduation tu/e), Mechanical Engineering, May 2019. Proefschrift.

[19] Martijn Wismans, Sten J. J. van den Broek, Lambèrt C. A. van Breemen, Leon E. Govaert, and Tom A. P. Engels. Micromechanical analysis of age-induced strength reduction in a multiaxially loaded short-fiber reinforced thermoplastic. *Journal of Applied Polymer Science*, 140(41):e54512, 2023. doi: https://doi.org/10.1002/app.54512. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/app.54512.

[20] Fadi Aldakheel, Elsayed S. Elsayed, Tarek I. Zohdi, and Peter Wriggers. Efficient multiscale modeling of heterogeneous materials using deep neural networks. *Computational Mechanics*, 72(1):155–171, Jul 2023. ISSN 1432-0924. doi: 10.1007/s00466-023-02324-9. URL https://doi.org/10.1007/s00466-023-02324-9.

[21] Zeliang Liu, C. T. Wu, and M. Koishi. Transfer learning of deep material network for seamless structure–property predictions. *Computational Mechanics*, 64(2):451–465, Aug 2019. ISSN 1432-0924. doi: 10.1007/s00466-019-01704-4. URL https://doi.org/10.1007/s00466-019-01704-4.

[22] Alexander Fuchs, Yousef Heider, Kun Wang, WaiChing Sun, and Michael Kaliske. Dnn2: A hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions. *Computers Structures*, 249:106505, 2021. ISSN 0045-7949. doi: https://doi.org/10.1016/j.compstruc.2021.106505. URL https://www.sciencedirect.com/science/article/pii/S0045794921000274.

[23] J. Bear. *Dynamics of Fluids in Porous Media*. American Elsevier Publishing Company, New York, 1972.

[24] E.A. de Souza Neto, D. Peric, and D.R.J. Owen. *Computational Methods for Plasticity - Theory and Applications*. A John Wiley and Sons, Ltd, Publication, 2008.

[25] N. Kovács. Physically recurrent neural networks for cohesive homogenization of composite materials. Master's thesis, TU Delft, 2023.