Chair of Applied Mechanics
Department of Mechanical Engineering
Technical University of Munich

A Neural Network Approach for Component-Level Reduction of Numerical Rubber Mounts

Semester Thesis

an der Fakultät für Maschinenwesen der Technischen Universität München

Betreut von Univ.-Prof. dr.ir. Daniel J. Rixen
Francesco Trainotti, M.Sc.
Chair of Applied Mechanics

Eingereicht von Batuhan Sancak
Am Schaeferanger 9
85764 Oberschleißheim

Eingereicht am Garching, den 25. Februar 2022

Contents

List of Figures

Abbreviation

ANN	Artificial Neural Networks
API	Application Programming Interface
CNN	Convolutional Neural Networks
CPU	Central Process Unit
DL	Deep Learning
DoF	Degree of Freedom
FEA	Finite Element Analysis
FEM	Finite Element Method
FFN	Feed Forward Networks
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
Hz	Hertz
LSTM	Long-Short Term Memory
MBS	Multibody Simulation
ML	Machine Learning
N	Newton
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks

Chapter 1

Introduction

1.1 Motivation

Rubber components find an application in many different fields and industries such as automotive, aerospace, medical engineering, commodity products and heavy machinery. Specifically in automotive industry, rubber components have an important role for the suspension systems of many different vehicles like cars or motorcycles. The preference of rubber components is mostly due to the good damping capability which they possess.

A rubber material exhibits a complex mechanical behavior due to the combination of different response dependencies such as load amplitude, frequency/rate, loading history of the material and also environmental conditions like temperature. This complicated behavior stems from the characteristic of rubber, or generically polymer, materials. A rubber material behavior can be described or modeled with the combination of the following properties:

- Elasticity
- Hyper-elasticity
- Elastoplasticity
- Viscoelasticity
- Viscoplasticity

During the research and development activities in different industries, finite element analysis (FEA) and multi-body simulations (MBS) are commonly used practices for a high fidelity, robust solution. In these simulation methods, modeling a rubber behavior and obtaining the correct result is an expensive and cumbersome process both in terms of time and available computational sources. Especially in a finite element model with high number of degree of freedom (DoF), the complicated nonlinear model of rubber material can make the computation pace very slow. To reduce this complexity, a neural network surrogate model can be constructed for representing the material model of the rubber component. Once this surrogate model is implemented in the whole finite element or multi-body model, computation cost of the whole model will be lower as neural network model prediction takes much less time than a multiphysics simulation result.

Deep learning is a field under machine learning that is utilized to learn relatively more complex, nonlinear functions and relations, where classical basic machine learning algorithms can not learn. Deep learning is in basic a neural network architecture that consists of an

input, an output and at least one hidden layer which gives the name deep learning to the model. Layers are formed of neurons that help learning the functions. A neural network model can have different types such as feed forward network (FFN), recurrent neural network (RNN) or convolutional neural network (CNN) that serve different purposes in different applications.

By choosing a good neural network design with the correct parameters and enough amount of data to train the model, full high-fidelity physics simulations of the rubber component can be replaced with this deep learning model. Main challenges to form the model are obtaining plenty of simulation data for training, as the accuracy and high fidelity of a deep learning model is very highly dependent on and mostly proportional to the amount of available data, and also finding a good combination of hyperparameters.

1.2 Goal of the thesis

The purpose of the thesis is to create a surrogate neural network model that can predict the physical behavior of a rubber component in terms of force/displacement input-output behavior. To do this, a finite element model has to be created with the correct material characteristics and has to be simulated for different loading combinations, so that these results can be used as training set for the deep learning model. Some of these simulations will also be used to test the predictivity/usability of the constructed deep learning model.

1.3 Thesis outline

The thesis is structured as follows: In chapter 2 the general theory of rubber material behavior and deep learning is introduced. The generation of the finite element simulations in ABAQUS to obtain the data for the deep learning models is explained in chapter 3. After that, chapter 4 explains how the surrogate neural network models are prepared to predict the physical behavior of the simulated rubber component and the results of each surrogate model are also illustrated. Finally in chapter 5, conclusive remarks along with suggestions on possible future work are provided.

Chapter 2

Theory

In this chapter, a theoretical basis of rubber material modeling approaches and deep learning algorithms will be given. The theory introduced here will be used during the rest of the thesis.

2.1 Rubber Material Model

Main properties of a rubber material with their formulations are explained first in section 2.1.1. After this, simple Maxwell model, Kelvin model are explained in section 2.1.2 and section 2.1.3 respectively. Finally, a generalized model principle is discussed in section 2.1.4

2.1.1 Basic Rubber Modeling

Rubber materials are capable of undergoing relatively very high loads without experiencing any plastic deformation. This points out to the elastic characteristic of a rubber which will be explained later under the term hyperelasticity. Rubber materials also possess a history dependency in their responses due to the chain structures of the polymers. This characteristic is modeled with viscoelasticity.

Over the history, many basic and more advanced models were developed to obtain a correct rubber behavior. In these models, elasticity property of the rubber is modeled with a spring which could be a linear or nonlinear elastic spring depending on the rubber behavior. Moreover, viscoelasticity is characterized by a rate-dependent viscous damper additional to a spring element in the material model. These damper elements and spring elements can be combined in different configurations and quantities to obtain a more complex or generalized rubber model. Frictional elements can also be added for modeling the plasticity behavior. A one dimensional simple model depiction with elastic, viscous and plastic properties can be seen in fig. 2.1. Spring in this model can be chosen as linear or nonlinear elastic based on the needed rubber behavior.

The essential rubber material properties to be modeled are:

1. Elasticity.
2. Hyperelasticity.

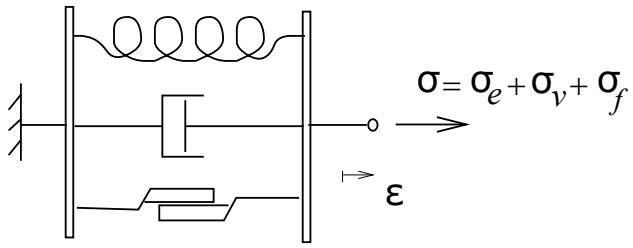


Figure 2.1: One dimensional simple elastic, viscous, plastic rubber model [10].

3. Viscoelasticity.
4. Elastoplasticity.

Elasticity

A linear elastic material follows the Hook's Law, which means that there is a direct linear relationship between strain and stress and this relationship is expressed via Young's Modulus. In one dimensional material models, linear elasticity can be simply modeled with a linear spring element. The material does not show irreversible deformations after the load is removed.

Hyperelasticity

Rubber materials can experience very large elastic deformations during operation without showing any plastic damage, especially when they are elastomere type of polymers. Often, a linear elastic model is not the correct way to model the rubber as the elastic deformation at very high elongations is not linear any more. This behavior is called hyperelasticity and it demonstrates the static/relaxed amplitude dependent property of the rubber material [10]. Hyperelastic behavior is formulated with the strain energy potential which is equal to the strain energy stored inside of a unit volume of the material. It is defined as a function of the strain at a specific point in the material [17]. Volume refers to the inital volume. Required parameters for modeling the hyperelasticity have to be determined via real life experiments.

In the literature there are various strain energy potential models which are of first, second or third order [17]. Some of these models are:

- Mooney-Rivlin form.
- Neo-Hookean form.
- Ogden form.
- Polynomial form.
- Yeoh form.

Based on the available experimental data and considering the fact that the rubber material being modeled is approximately incompressible, a reduced polynomial form modeling was used in this thesis which is a modified version of the polynomial form model. Strain energy potential of the reduced polynomial form is [17] :

$$\mathbf{U} = \sum_{i=1}^{N} C_{i0}(\overline{I_1} - 3)^i + \sum_{i=1}^{N} \frac{1}{D_i}(J^{el} - 1)^{2i} \tag{2.1}$$

where U is the strain energy in a unit volume, N is a material constant, $\mathbf{C}_{i0}$ and $\mathbf{D}_i$ are material parameters depending on temperature, $\overline{\mathbf{I_1}}$ is the first deviatoric strain invariant formulated as:

$$\overline{\mathbf{I_1}} = \overline{\lambda_1}^2 + \overline{\lambda_2}^2 + \overline{\lambda_3}^2 \tag{2.2}$$

where the deviatoric stretches $\overline{\lambda_\mathbf{i}} = \mathbf{J}^{\frac{-1}{3}}\lambda_\mathbf{i}$, J is the total volume ratio, $\mathbf{J^{el}}$ is the elastic volume ratio and $\lambda_\mathbf{i}$ are the principal stretches. The initial shear modulus and bulk modulus are respectively defined as:

$$\mu_\mathbf{0} = 2C_{10}, \quad \mathbf{K_0} = \frac{2}{D_1} \tag{2.3}$$

Viscoelasticity

Rubber materials do not behave like perfectly elastic materials which are modeled with a simple linear/nonlinear spring. In a perfectly elastic material, there is not time dependency between stress and strain. It means that, it does not matter if a force is applied to an elastic material very slowly or very fast, the deformation or strain response will be the same. Instead, rubber materials show a time dependency in their stress and strain relationship [19]. This dependency is explained by viscoelasticity, or depending on the material, it could also be viscoplasticity, which is not discussed in this section.

When a viscous material is first loaded, and then unloaded, loading and unloading curves do not overlap with each other like elastic materials, instead they form a hysteresis loop [11]. This behavior shows that there is some energy dissipated in this process via heat transfer due to the deformation and movement of polymer chains which gives a damping characteristic to the material [19]. There is also a rate dependency between stress and strain parameters of the rubbers. If a load is applied more quickly, the material demonstrates a stiffer behavior which means a smaller deformation. On the contrary, when the load is applied slowly, the material reacts in a softer way which means a bigger deformation response to the same force amplitude [11]. These features of a viscoelastic material can be seen in fig. 2.2.
As seen on figure 2.2, viscoelastic material can also have a permanent deformation based on its preload history.

Related to their time dependencies, viscoelastic materials show creep and stress relaxation behaviors [16]. If the material is loaded under a constant load over the time, its deformation will keep increasing until converging a limit value. This is the creep behavior. If the material is loaded under a constant deformation, the stress over the material will decrease towards a limit value. This is the stress relaxation behavior. This behavior is a result of the rearrangement of the molecules [10]. These behaviors based on the loading can be seen in fig. 2.3 and fig. 2.4 respectively.
Creep behavior is formulated with creep compliance function J(t), and stress relaxation is formulated with relaxation modulus E(t) [13] . These both parameters are defined as :

$$\epsilon(\mathbf{t}) = \sigma_\mathbf{0} J(t), \quad \sigma(\mathbf{t}) = \epsilon_\mathbf{0} E(t) \tag{2.4}$$

with creep compliance being strain due to stress and relaxation modulus being stress due to strain.

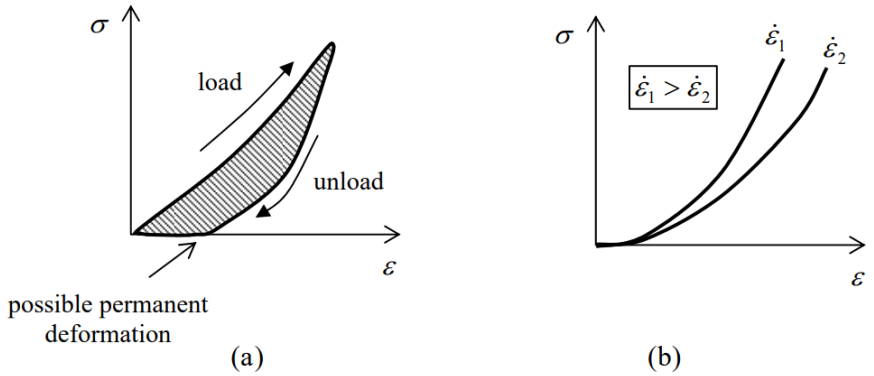


Figure 2.2: Viscoelastic behavior under tensile loading, (a) loading/unloading curves with possible permanent deformation, (b) deformation at different rates [11].

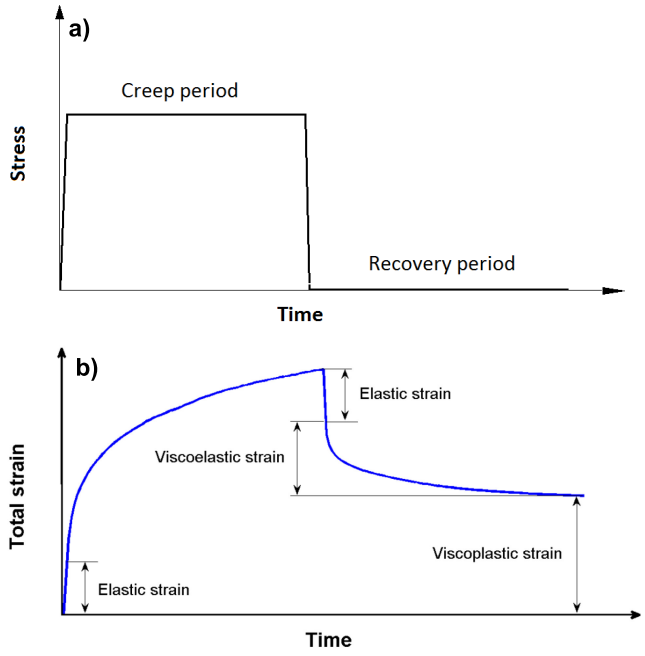


Figure 2.3: Typical strain response to a creep recovery test with a possible plastic permanent deformation [16].

2.1.2 Maxwell Model

When a spring element representing the elasticity and a dashpot representing the viscosity are connected in series, a basic Maxwell model is obtained as seen in fig. 2.5.
The total strain in the model (ϵ) can be divided into two seperate parts which are ϵ_1 for the spring strain and ϵ_2 for the dashpot strain [11]. As the stress on both serial elements has to be the same, it can be written:

$$\epsilon_1 = \frac{1}{E}\sigma, \quad \dot{\epsilon_2} = \frac{1}{\eta}\sigma, \quad \epsilon = \epsilon_1 + \epsilon_2 \tag{2.5}$$

This equation can be written in a standard form with the left side with stress and right side with strain. In this way, final equation of Maxwell model becomes:

$$\sigma + \frac{\eta}{E}\dot{\sigma} = \eta\dot{\epsilon} \tag{2.6}$$

Even though Maxwell model can simulate the creep behavior at some extent, it is still not a perfect model. When a rubber component recovers from creep, Maxwell model demonstrates

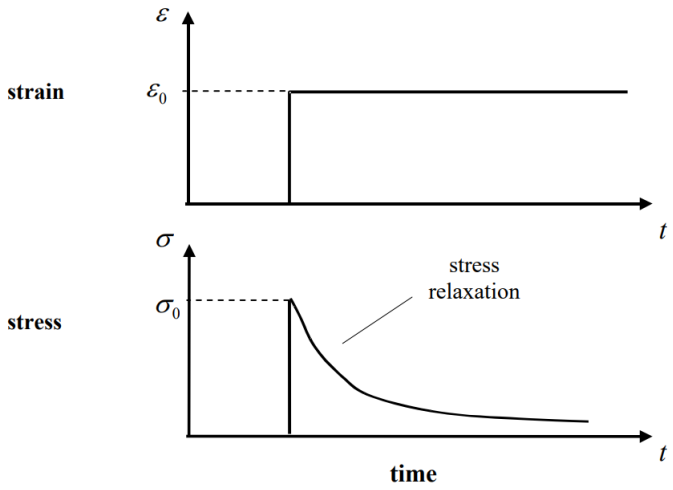


Figure 2.4: Stress response under stress relaxation test [11].

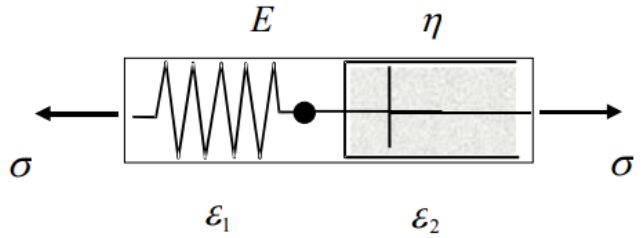


Figure 2.5: Basic Maxwell model [5].

only the elastic recovery part and the plastic permanent strain(if there is any) [11]. Plastic recovery is not estimated by Maxwell model [1].

2.1.3 Kelvin Model

In the Kelvin model, previous spring and dashpot elements are connected in parallel. This configuration implies that strains on both elements are the same and the total stress is divided between the spring and dashpot [11]. A basic Kelvin model can be seen in fig. 2.6. Based on

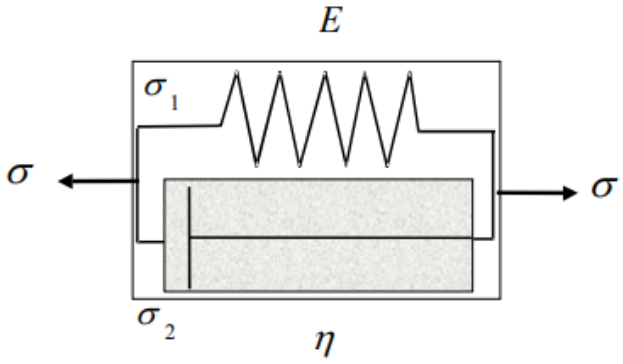


Figure 2.6: Basic Kelvin model [5].

[1] More detailed explanation for the Maxwell model can be found in the appendix of [10].

the law above, the equation for Kelvin model can be written as:

$$\epsilon = \frac{1}{\mathbf{E}}\sigma_1, \quad \dot{\epsilon} = \frac{1}{\eta}\sigma_2, \quad \sigma = \sigma_1 + \sigma_2 \tag{2.7}$$

σ_1 is the stress of the spring and σ_2 is the stress of the dashpot. The standard form of the Kelvin model can be written as below:

$$\sigma = \mathbf{E}\epsilon + \eta\dot{\epsilon} \tag{2.8}$$

In a creep response, Kelvin model can predict the elastic and plastic recoveries, but when the material is suddenly loaded with stress, the spring will not response immediately. This is because the dash-pot will hold the spring back and take the initial stress. In this way, the initial response to a sudden load will not be an immediate increase in strain, instead a linear curve with slope [2]. For an instantaneous strain response, an infinite amount of stress has to be applied [11].

2.1.4 Generalized Models

As explained above, simple standard Maxwell and Kelvin models are not perfectly adequate to express the behavior of a viscoelastic rubber material even though they are good starting points. To increase the capability of these material models, one can use more generalized models. These generalization can be made by using a combination of simpler models like seen in fig. 2.7 which is a generalized Maxwell model where many Maxwell elements are connected in parallel. There is also a parallel isolated spring in the generalized Maxwell model. This spring can be linear or nonlinear elastic which gives the linear elasticity or the hyper-elasticity property to the modeled material. Parallel Maxwell elements represent viscoelasticity. Eventually, friction elements can be added to these models to simulate the plastic effects. The more elements a generalized model has, the more correct estimation

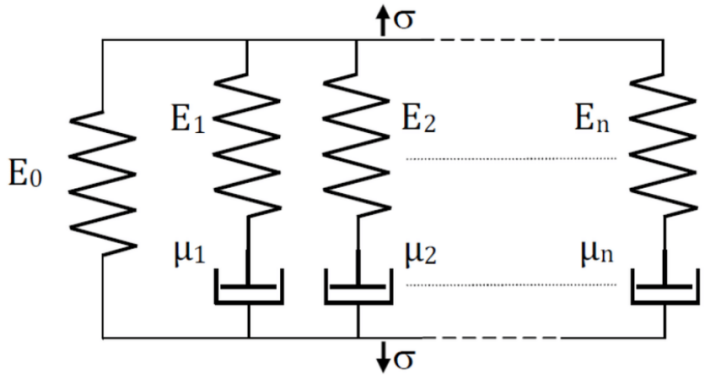


Figure 2.7: Generalized Maxwell model [1].

for the behavior of the material is generally obtained. However, this increase in element number leads to a more complex model with many material parameters which one needs to determine. This increases the effort for experimentation [11]. Relaxation and complex modulus of the material can be calculated via Prony series.

[2] More detailed explanation for the Kelvin model can be found in the appendix of [10].

2.2 Neural Networks

A machine learning(ML) model is able to predict an output, which could be a numerical value or a classification result, based on the input data provided. ML models are enabled to do this prediction by providing enough amount of data to train them so that the model can build a mapping function between input and outputs [8]. As the amount of data in every branch keeps increasing day by day, ML is able to find application in many diverse areas including engineering.
Deep learning(DL) is a subclass of machine learning which is a model consisting of an input, an output and at least one layer in between that gives the name deep learning. These layers contain neurons that interact with the neurons of other layers. DL is especially useful when the problem is very nonlinear because every neuron or node in layers can have nonlinear activation functions that can predict even very nonlinear and complex input-output relations, whereas classical ML algorithms can demonstrate a limited capacity [12]. DL models also tend to have a better accuracy proportional to increasing data no matter how much data is provided. However, ML models can not increase their accuracy after a level even though one provides more and more data because of the limited capability of classical ML [15]. It should also be noted that, more data means a longer training process due to the higher computation effort.

There are different type of artificial neural networks(ANN) which are used depending on the problem type. The basic and the most commonly used type is feed forward networks(FFN) whose design can be seen in fig. 2.8.

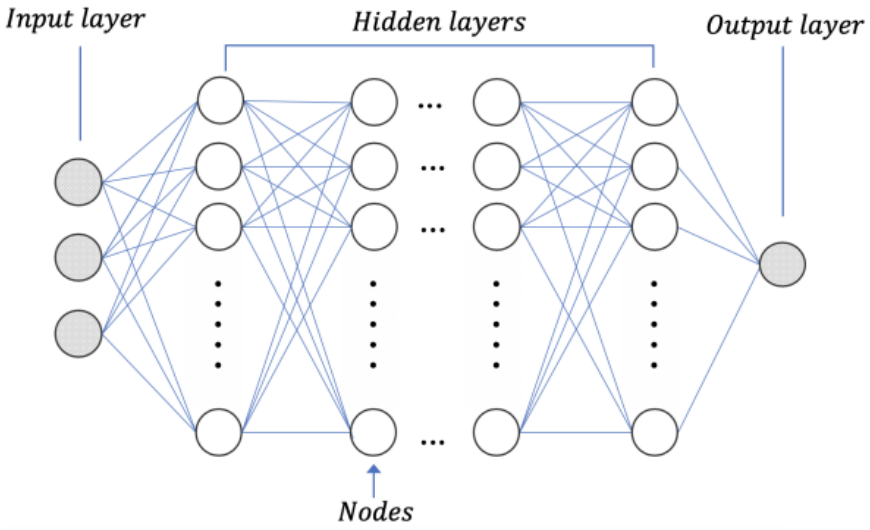


Figure 2.8: FFN architecture with hidden layers and nodes [3].

Input and output layers have as many nodes as the input and output parameters. Number of layers between them and their node quantities are determined by trial and error which can be done manually or via search algorithms. This process is called hyper parameter optimization. Hyper parameters are the characteristic features of an ANN model which are not changed while the model is training or predicting the results and they are crucial for the accuracy or the effectiveness of a DL model [8]. Some important hyper parameters are:

- Number of hidden layers.

- Number of nodes in a hidden layer.
- Number of total iterations(epoch).
- Batch size.
- Activation functions of the nodes in a layer.
- Optimizer.
- Test set size.

FFN architecture explains the fundamental way of working of ANNs. Starting from the input layer until the output layer, each neuron's output in a layer is multiplied with a weight constant and added a constant bias value and after these operations, it becomes an input value for every node in the next layer. A node in a layer takes input from each node in the previous layer. Weight and bias values are different between every single node pairs.
Nodes contain activation functions which enable them to generate nonlinear relations by converting their input to a different output via activation. There are different types of activation functions like sigmoid, hyperbolic tangent or rectified linear unit(ReLU) [8]. ReLU is currently the most used one and its definition can be seen in in fig. 2.8 [8]. In an ANN

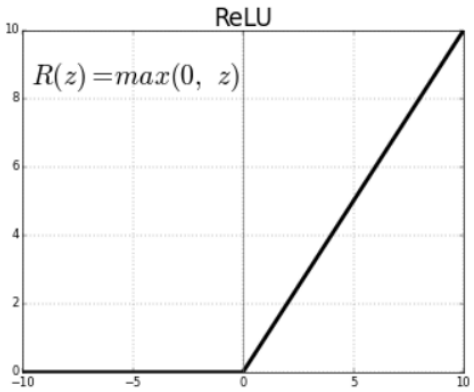


Figure 2.9: ReLU activation function.

model, weight and bias values between nodes are updated at every epoch, which is one pass of the whole input dataset. This pass is realized by a back propagation algorithm which uses an optimizer to minimize the loss function. In the model, loss function could be for example a mean squared error between the true(expected) output values and the output values predicted by the model. The optimizer algorithm could be gradient descent, stochastic gradient descent, Adam or momentum [15]. This iterative optimization process teaches the model the connection between input-output data and finally determines appropriate values for weights and biases. Once the model is trained, it is ready to predict the data that it has never seen before. This data group is called test data. Train and test data percentages of the whole available data set is also an important parameter [8].

Another type of ANN is recurrent neural networks(RNN) which are used to predict sequence type of data. This sequence could be a numeric time series or even a sentence. Many translation apps use RNN technology for natural language processing(NLP) models. While predicting the output, a RNN networks does not only take input from the current time step, but also utilizes the information from the inputs of the previous time steps as well [8]. This is managed via their memorizing capability that enable them to store the data from previous

input steps as well. Both input and output data of RNN are sequences or time series. A basic RNN architecture can be seen in fig. 2.10 with every blue box being an RNN cell with many neurons. Each time step's data goes through an RNN cell and gives output to the next time step's input. RNN models take longer computational time compared to FFNs due to extra

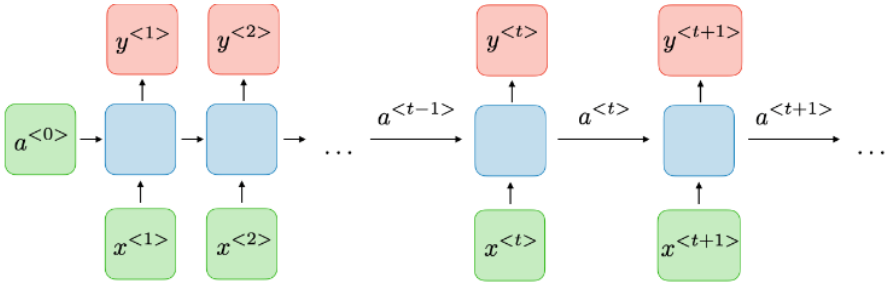


Figure 2.10: A basic RNN model architecture [4].

effort to store memory data and giving a feedback to the next time steps [8].
RNNs have a problem with longer time series. Due to the exponential gradients, RNN may not be able to hold information about long time dependencies as the gradient might get too big or too small very quickly during the back propagation. These problems are called exploding gradient and vanishing gradient [8]. Exploding gradients is more common in RNN applications with very long time series. This is because the back propagation goes through from the last time step to the first one and gradients are multiplied with each other at every step towards the first one. This situation can easily lead to an unstable gradient problem for the earlier time steps, which means the exploding of it. The other common reason for the exploding gradient is a bad weight initialization of the DL model. In that case, initial weights can lead to very high losses after each epoch that cause the exploding gradients. Vanishing gradient problem usually stems from the activation functions of the neurons. Activations like sigmoid function have almost zero gradients for very big(positive infinity) and very small(negative infinity) values. This leads to very small gradient magnitudes at every epoch and the optimizer can not find the solution. To solve these problems, two new variations of RNNs are developed which are gated recurrent unit(GRU) and long-short term memory(LSTM). As LSTM is a generalized version of GRU, it will be explained in the following section.

2.2.1 Long-Short Term Memory

As a solution to the problem of classical RNNs not being able to estimate the current output if an input from a very long time step before affects it, Hochreiter and Schmidhuber have developed the new architecture called LSTM. Differently from the classical RNN, LSTM cells have 3 gates called forget, input and output that regulate the data flow between time steps [9]. With these gates, an LSTM model can understand if a parameter from a very early time step has a significant effect on the output of the future time steps and as a result, information from earlier time steps can be given more weight to be stored. In the same way, less important parameters tend to be forgotten by the model via these gates. All these gates in a cell interact with inputs and each other via activation functions and mathematical operators as can be seen in fig. 2.11.
Both classical RNNs and also LSTMs can have bidirectional information flow, which means the output of the current state takes input from both previous and next time steps [8].

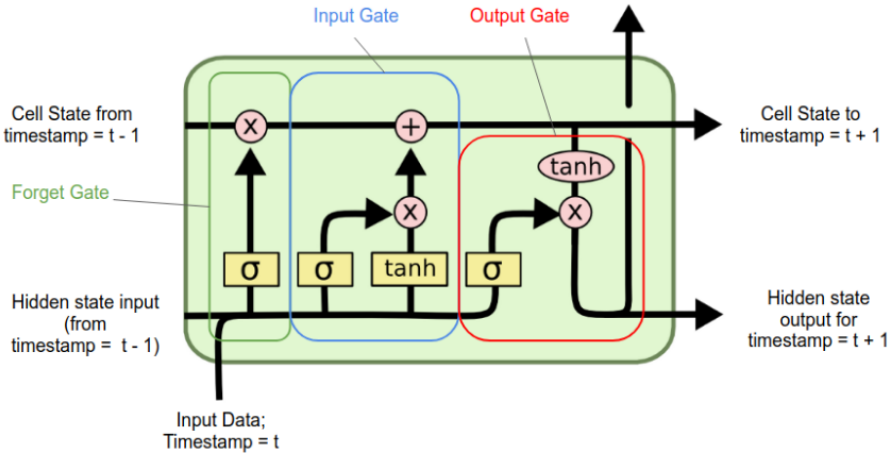


Figure 2.11: Structure of a single LSTM cell [18].

Chapter 3

Data processing and generation

In this chapter the steps to prepare a finite element simulation and automatically run hundreds of simulations with different loading conditions are explained. After that, another step to automatically post-process the simulation results for using them in the deep learning models later is described. Setting up a correct and realistic ABAQUS model is discussed in section 3.1, Python scripting for automatic analysis run and post-processing the results is explained in section 3.2 and finally the evaluation of the obtained simulation results is made in section 3.3.

3.1 Building the finite element model

The FEM should simulate the material properties realistically, so that it can be a good base for a realistic DL model too. As the focus of this thesis is a correct material model, a minimal FEM with relatively less elements is enough for this purpose. A small size FEM model has the advantage of less computational time per simulation and an easy troubleshooting. Apart from that, any external geometric effects/distortions can be excluded from the pure material response by using a minimal model.
A simple cube model with 1000 first order hexagonal elements was prepared. The cube was fixed in all directions from all of its nodes at the lower side and the nodes on the upper side are hold together with a kinematic coupling element. The loads are given via the kinematic coupling master node. The prepared model can be seen in fig. 3.1. The rubber material

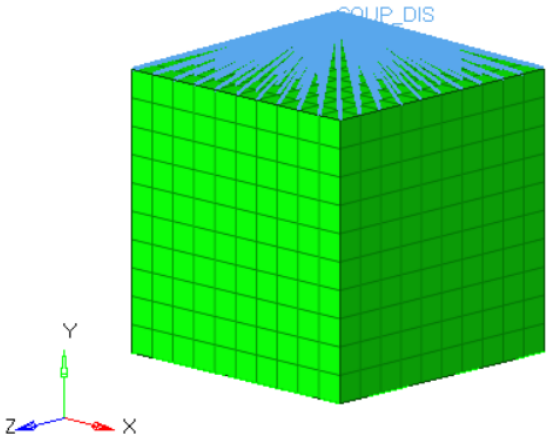


Figure 3.1: Finite element model to be used in the simulations

of the cube is hyper-elastic and viscoelastic. Reduced polynomial hyper-elasticity and time

dependent viscoelasticity material cards were used in ABAQUS to model the material, based on the previous work.
As loading condition, two different cases are applied to the component, namely a constant force magnitude loading and a periodic displacement loading.

3.1.1 Constant magnitude load

The rubber material is viscoelastic and due to this characteristic it shows creep behavior under constant load. So under a constant force(or stress), displacement response(or strain) is expected to keep increasing from the initial value until converging to a constant value by time. In this simulation case, 150 different force magnitudes are applied to the component with each magnitude being a separate simulation and displacement results are recorded. The minimum applied force is 0.1 Newton(N) and the maximum force is 400 N. This range is divided into 150 equal increments and each incremental value is applied as force in a separate simulation. 150 simulations are to be run in both normal and tangential directions to the loaded surface of the component to see the responses in both cases. So in total, there are 300 simulations. The type of the solver used in ABAQUS is dynamic implicit solver. Each time step is 0.5 seconds long, where the whole simulation runs for 100 seconds with total 200 time steps.

3.1.2 Periodic load

Rubber materials which are viscoelastic show rate dependency. If a load is applied in a faster way, the response of the material will be stiffer and a slower load application will get a softer response. When a periodic load is applied at a low frequency(low rate) and a high frequency(high rate), the responses for the same load magnitude will not be the same. There will also be a time shift between load and response curves. In the periodic load case simulation, a displacement load is applied to the component at 30 different frequencies and with 15 different displacement magnitudes, which means there are 450 different simulation cases in total with different load-frequency combinations. Applied load magnitude range is between 5E-5 mm and 0.1 mm, whereas the applied load frequency range is between 0.1 Hertz(Hz) and 1000 Hz. As result, reaction forces are recorded. Loads are applied in the normal direction to the loaded surface. Type of the solver used in ABAQUS is dynamic implicit solver. As simulation time, each simulation was performed for three whole periods. As the load frequencies are different, total length of the simulation(3 whole periods) in terms of seconds also varies. Length of a single time step is adjusted in such a way that 1 whole period has 100 time steps in total. Again, as the frequencies are different, time step lengths also differ.

3.2 Automation of the simulation

ABAQUS provides a scripting interface to its user which is an application programming interface(API) for interacting with ABAQUS data and models. Scripting interface of ABAQUS is Python based and supports object oriented programming in this way. Scripts that interact with ABAQUS models are also written in Python. By using this ability, many ABAQUS tasks, especially the ones that repeat themselves at some stage, can be automatized via Python

scripts. The simulations of this thesis require many repetitive simulations with different parameter combinations, hence the need of a proper process automation.
After having run all the simulations, data need to be post-processed in a way that they can be used compatibly within the DL. ABAQUS API with Python comes into action also at this step. A Python script can be written to read the needed output data and pre-process it for the DL model.

3.2.1 Parametric simulation scripting

Computing the response of a model to various loading conditions require a parametric study. ABAQUS supports a user written Python scripting to automatize these simulations. For the simulations in this thesis, displacement load magnitude and the frequency at which the load is applied in the periodic loading case are parameterized in the ABAQUS input files and a Python script was developed to create a combination matrix from load magnitudes and frequencies(for the simulations case with constant load, only the load was parameterized) and run ABAQUS simulations with all these combinations.

3.2.2 Scripting for post-processing

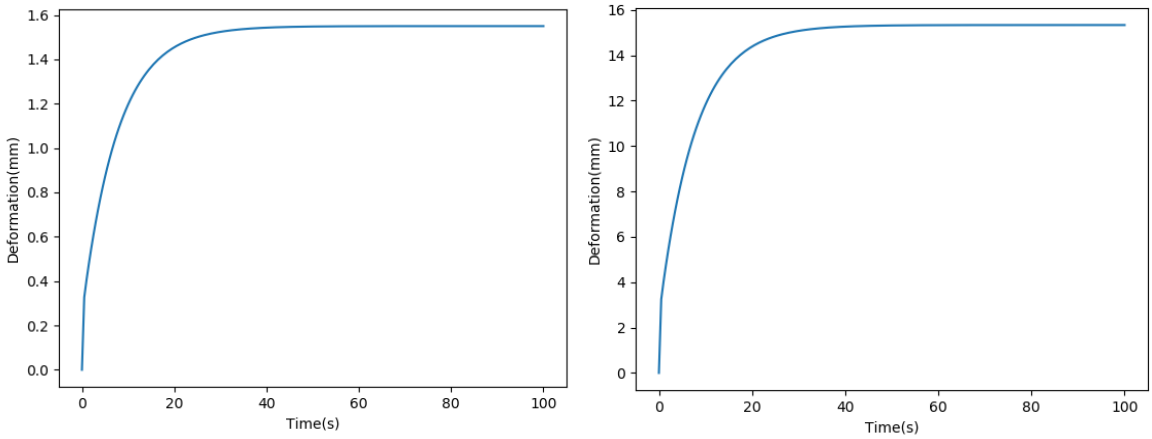
After the simulations are completed, reaction forces or displacements based on the load case considered are supposed to be collected from all of the result files. Python scripting interface of ABAQUS enables to interact with the model output data as well for reading, performing mathematical operations or manipulating and writing into different types of files. A Python script was written also for these steps to get the output data from the result file, converting it into tabular data to be compatible with the DL models in the next steps.

3.3 Evaluation of the results

When all the simulations are done for both constant load magnitude and periodic loading cases, some result checks are supposed to be done to make sure that obtained results can reflect the real life behavior of a rubber material like hyper-elasticity, rate dependent viscoelasticity and creep. Results of the constant magnitude load case and periodic loading case are examined in section 3.3.1 and section 3.3.2.

3.3.1 Constant magnitude load results

Under constant loading, the rubber materials are expected to show creep behavior due to viscoelasticity. All the simulations for this load case were run for 100 seconds of loading at a constant amplitude. Reaction force as output data was collected for every 0.5 seconds increment. As an illustrative sample, the result of the case with 400 N loading in normal and tangential direction to the loaded surface can be seen in fig. 3.2 (a) and (b) respectively.
As can be seen in fig. 3.2, creep response result from the simulation looks as expected. Deformation starts with a rapid increase at the beginning of the loading, then its increase rate decreases gradually and towards the end of the simulation, it converges to the final deformation value. It can also be observed that the material behaves much more stiffly against a normal loading than a tangential loading.



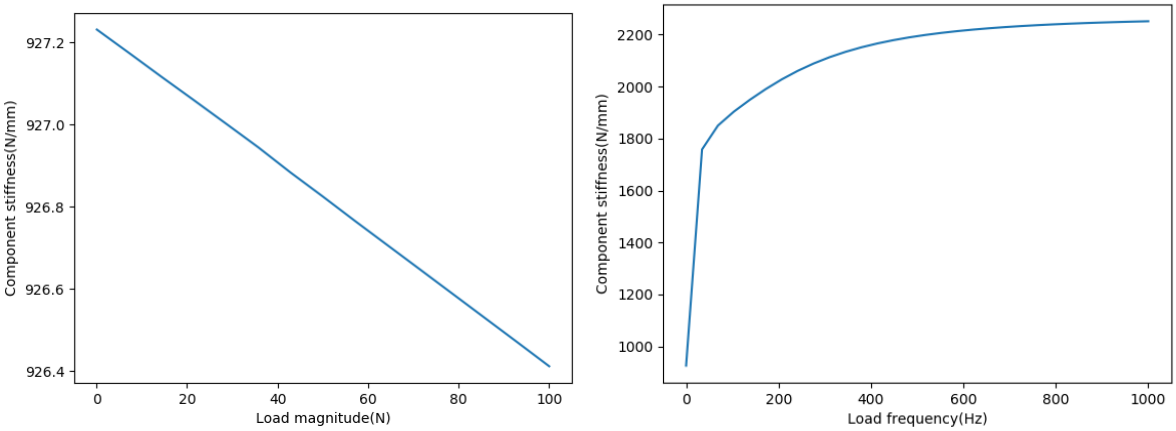
(a) Creep response to 400 N in normal direction

(b) Creep response to 400 N in tangential direction

Figure 3.2: Creep response of the rubber to different loading directions.

3.3.2 Periodic loading results

Under a periodic loading, the response of the rubber material changes based on the frequency at which the load is applied as it is viscoelastic. With the increasing frequency which means the loading rate or speed is higher, it will stiffen. Also due to hyper-elasticity, its response is expected to be less stiff with increasing load magnitude. In fig. 3.3 (a), stiffness response(applied force divided by the deformation) of the rubber material in the simulation results based on the increasing applied force can be observed while the frequency at which the load is applied is kept constant. Even though all the simulations for the DL model were made by applying deformation load, the simulations in fig. 3.3 (a) are performed by applying force load. This is because the stiffness calculation simulations were performed earlier than the DL model simulations with different models. Moreover, in fig. 3.3 (b), viscoelasticity response of the rubber can be seen based on increasing load frequency while the magnitude of the load is kept constant. Load is applied as sine wave and its direction is normal to the loaded surface of the cube geometry.



(a) Hyper-elastic response

(b) Viscoelastic response

Figure 3.3: Response of the rubber to periodic loading.

As can be seen in fig. 3.3, the viscoelastic response of the rubber is obvious as the stiffness of the component keeps increasing with higher frequencies when the force magnitude is kept

constant. On the other hand, stiffness response in the hyper-elastic graph is barely changing with increasing load when the frequency is kept constant. That is due to the material characteristic which is quite close to a linear elastic behavior at relatively low-middle magnitude loads. Hyper-elasticity comes into effect at much higher loads. In the loading cases for the DL simulation, applied loads are between low and middle ranges as very high loads lead to excessive deformations and convergence issues with the available material model, even though nonlinear geometry solving option is activated in ABAQUS.
Apart from the properties shown above, the damping behavior of the component with changing frequency can also be obtained, even though it is not included in the thesis. As the phase shift between load and response is known from the FEM results at each frequency, the loss and storage modulus of the component can be calculated. Furthermore, loss factor can be obtained by these values which quantifies the damping capability of the rubber.

All the simulations for the periodic loading case have been performed for 3 whole periods time duration. The output from ABAQUS was requested for 100 time points for each period to obtain a high resolution result, especially in order to get the phase shift between the load and response at higher frequencies. This is because the time steps for high frequency simulations are too small and the loading time periods for these cases have to be split into very small time increments. In our case, 100 data points per period is enough to catch the phase shift at 1000 Hz which is the maximum frequency. As all the simulations are run for 3 periods, they all have 300 time steps(data points) in total.

Chapter 4

Deep learning modeling

The previous chapter explained how the required data was collected to build and train the DL models. Data for the neural networks come from simulations in this thesis but in some applications the source could be experiments too even though it would be much more cumbersome and probably expensive compared to the simulation sourced data.
In this chapter, it will be explained how the DL surrogate models were built with the available data and also their results will be shown. Firstly in section 4.1 the DL model preparation for the creep simulations will be illustrated and the results of both FFN and LSTM architectures will be shown. After that in section 4.2 the modeling and results for the periodic load case are similarly presented. Finally in section 4.3 programming methodology of the DL models in this thesis is introduced.

4.1 Deep learning model for creep behavior

In this section it will be discussed which different DL models are built to predict the creep behavior of the rubber component. In section 4.1.1 the structure and building of the DL models are explained, the obtained results with validation are shown in section 4.1.2.

4.1.1 Preparation of deep learning models

For the neural network problems in this thesis two different types of architectures are compared which are FFNs and LSTM cells. The results arising from both these structures are compared to each other in the end.
An LSTM model was prepared to predict the creep results. Firstly, an isolated LSTM model was built for 150 creep simulations in the normal direction loading. This model used 120 of these simulation results as training data, and the rest 30 simulations were utilized as test data to validate the performance of the DL model compared to the full FEM results. First 60 of the 120 training data are from the simulations with a force magnitude range from 0.1 N to 158.4 N. The rest 60 training simulations are with a force magnitude range from 241.6 N to 400 N. Thus, the test data are the simulations between 158.4 N and 241.6 N. In this way, applied force magnitudes in the test data do not exceed the training data magnitudes. The reason for this adjustment is the known fact about LSTMs(and also most ML algorithms) losing prediction accuracy for the values out of the training range [7]. This is because an extrapolation is a challenge for a regression problem. However, as the loads that can come to a component in real life operations are pretty much known(especially the maximum force), this condition represents a good training practice between a minimum and a maximum value.

Input time series for the DL is the applied constant force, and the output is the corresponding computed deformation response.
As LSTM type, bidirectional LSTMs are used. Bidirectional LSTMs, as different from the normal LSTMs, use also the data at the future time steps in a time series while predicting the output for the current time step, whereas normal LSTMs use only the data from previous time steps as input. Even though bidirectional LSTMs might require higher training time, they usually exhibit a better accuracy [14]. As the LSTM architecture, sequence to sequence modeling approach with encoder decoder structure was used. In this structure the DL model is split into two parts which are encoder and decoder. Each of these sub parts include an LSTM cell. Encoder part takes the input as a whole time series, processes them and outputs a hidden result. This hidden result is hold in the memory and transferred to the second part which is decoder. After getting the original input and also this hidden output, decoder generates the final result of the model [2]. This result was demanded as a whole time series instead of one singular value as the problem in this thesis is predicting the output as time series. Encoder-decoder models enable the user to work with input series from different lengths if needed. An encoder-decoder architecture can be seen in the figure 4.1.

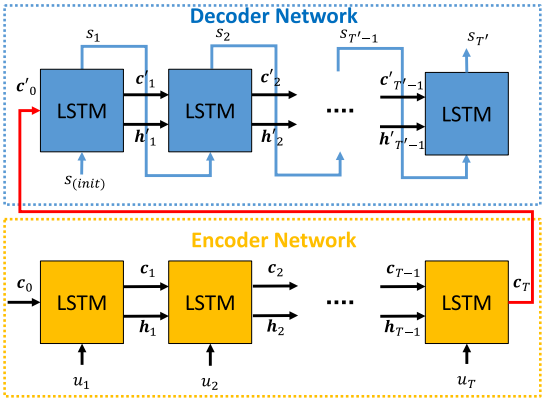


Figure 4.1: Encoder-decoder structure for sequence to sequence modeling

In this model, one bidirectional LSTM cell was used for encoder and decoder, which makes 2 in total. Each cell included 50 neurons. Input and output series, which are force and displacement, were scaled by using a minimum-maximum scaling, such that all the values lie in the range between 0 and 1. This normalization is performed by:

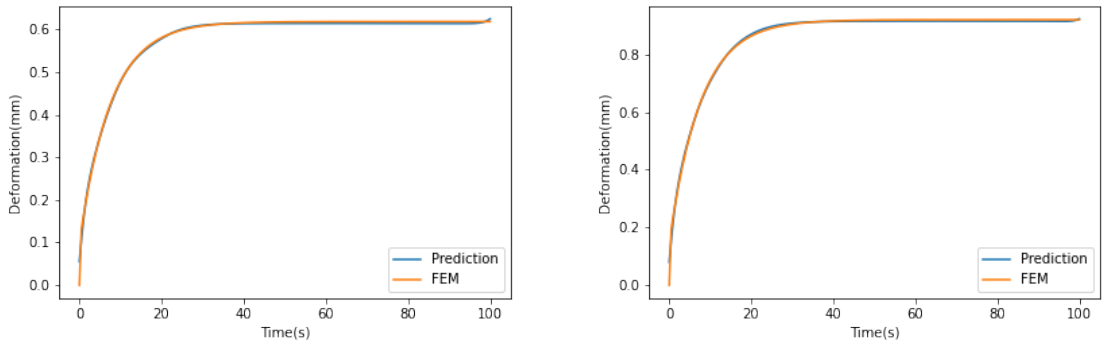
$$\mathbf{x}_{\mathbf{scaled}} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

where $\mathbf{x}_{\mathbf{scaled}}$ is the scaled value of the data and $\mathbf{x}$ is the original data. $\mathbf{x}_{max}$ and $\mathbf{x}_{min}$ are the maximum and minimum values in the whole dataset respectively. This scaling step is crucial for ML applications because if the input values have different scales as magnitude, this slows down the error minimization process in the training a lot, and the model may even never complete the training because of the failed gradient descent.
Input and output series in this model have 201 time points which are the simulation increments from ABAQUS. Each increment is equal to 0.5 seconds of simulation time which makes 100 seconds per series in total. Activation functions of LSTM nodes are rectified linear unit. Optimization algorithm is selected as Adam and the loss function is mean square error. Adam is an adaptive gradient descent algorithm, which is an extension of the stochastic gradient descent. It can update the learning rate for each epoch. Epoch number is selected as 500 iterations and batch size is 120 which is the whole training set size. Including all the training

samples in one batch is known as full batch training. Even though it usually takes longer time to train a model with full batch method compared to the mini batch, it has no randomness in an iteration and can find the best gradient descent path which makes it a good choice especially at relatively small data sets(e.g less than 1000 data size) [8].
Apart from the LSTM model, also an FFN model was prepared. However, FFN networks can not properly process time series data, instead they take single data value as input and also give single data as output. A FFN can work on the time series problems only when the change of a variable with time is defined with its derivative by time as an extra input/output feature. However, this approach is not implemented in this thesis.
For a creep loading, where the input series is a constant force during the whole time flow and output series consists of increasing deformation values against constant loading, FFN is not expected to deliver a correct prediction. This is because FFN principally applies a mapping between input and output values via weight factors, but in this problem the output values keep changing even though the input magnitude remains the same. FFN can not understand this behavior as it has no idea about time series. Many different output values corresponding to the same input force magnitude confuses the FFN structure. In spite of this, the FFN model was built anyway to illustrate this effect. Input and output values are not series anymore, instead they are 201 different single time points for each series. Like in LSTM, 80% of the data set is used as training and the rest remains for the test. The model consists of three hidden layer with 50 neurons each. Input-output scaling, optimizer and hidden layer activation functions are the same as the one used in the LSTM model. Epoch is set as 600 iterations and the batch size is 360.

4.1.2 Deep learning model results for the creep load case

Training of the LSTM model took 20 minutes. As the outputs of the model had been previously scaled between 0 and 1, they have been scaled again in a reverse way to their original values before plotting. In fig. 4.2, the results of the LSTM prediction and FEM simulations can be seen for the cases with 161.1 N and 238.9 N, which are the minimum and maximum load magnitude of the test batch. All the other results between them show very similar behavior and accuracy and for that reason they are not illustrated in the figures.



(a) Result for 161.1 N load

(b) Result for 238.9 N load

Figure 4.2: LSTM and FEM results for two different creep results.

LSTM model seems to predict the creep deformation result with very high accuracy for both force magnitudes. Furthermore the accuracy for the models with a load magnitude between these minimum and maximum values is also very high.
In fig. 4.3, the prediction of the FFN model can be seen for the 238.9 N load case.

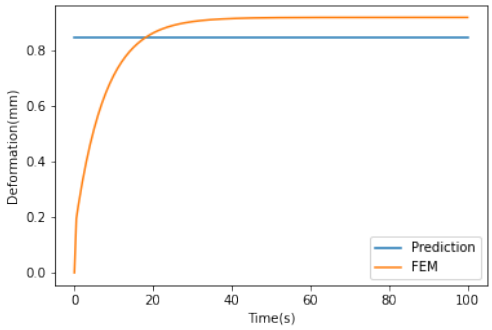


Figure 4.3: FFN and FEM results for 238.9 N load creep result

As foreseen before, FFN can not deliver a correct time series prediction. Instead of this, it gives an average value as output which it calculates from all the deformation values that start from 0 and keep increasing to the final magnitude for 238.9 N load.

Other 2 modeling cases and respective LSTMs are prepared apart from the LSTM model only for the loading in the normal direction.
The first LSTM model is built to predict the creep response for the forces that have higher magnitude than the ones used in the training of the DL model. In this manner, extrapolation capabilities of the LSTM model are tested for the out of range values. Hyper parameters of the model are the same with the ones used for the previous LSTM model, with the only difference being the selection of training and test batches. The 120 training data are the force magnitudes between 0.1 N and 319.4 N, whereas the test data is between 322.1 N and 400 N, which is out of the training range. Training of this model also took 20 minutes. Loads in these simulations are also in the normal direction. The case with the worst accuracy is the prediction of the 400 N load response whose results can be seen in fig. 4.4.

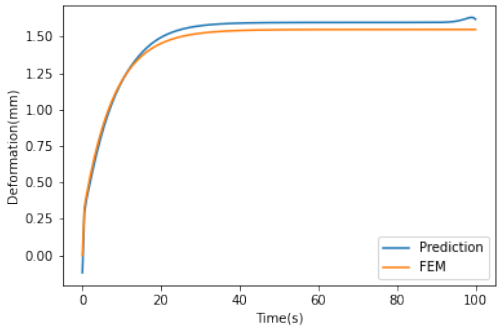
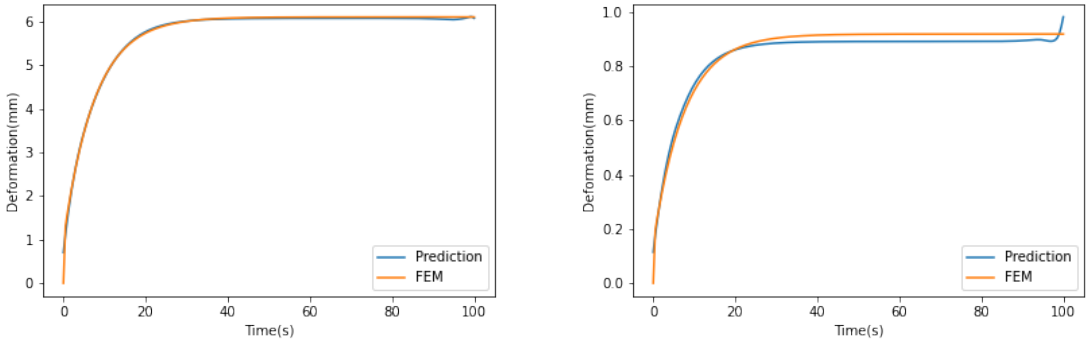


Figure 4.4: FEM and extrapolated LSTM results for 400 N load creep result

As fig. 4.4 shows, even the worst case scenario for predicting a result value which is relatively far from the training data set range turns out to be very accurate, a small deviation from the FEM results is only observed towards the end time of the loading.

The second LSTM model was prepared to include the behavior based on load direction. In terms of training data, this LSTM takes 120 of the simulations that are loaded in the normal direction to the loading surface and 120 from the simulations whose loading is in the tangential direction to the surface. Concerning the test data, 30 simulations for each loading directions are taken. The input to this LSTM model includes now 2 time series which are the

loads in the two directions. In order to include the loading direction feature in the LSTM, the time series for the unloaded direction is filled with zeros. For example, for a simulation that is loaded in normal direction, its tangential force component is written as 0 N for all the time steps in the input sequence, and for the tangential loads, the other way around. With this input data pattern LSTM can be aware of the directions of the loads. In fig. 4.5 the result of this LSTM model can be seen for two different load cases whose directions are different.



(a) Result for 161.1 N load in tangential direction

(b) Result for 238.9 N load in normal direction

Figure 4.5: LSTM and FEM results for two different creep results including the dependency on load directions.

According to fig. 4.5 the built LSTM model can successfully predict and discern a result based on the direction of the applied load.

4.2 Deep learning model for periodic loading

Different DL models with their hyper-parameters along with the data processing for the periodic loading response prediction are firstly discussed in section 4.2.1. After that, results of the different models are shown in comparison with the FEM validation data. .

4.2.1 Preparation of deep learning models

Also for the simulations with periodic displacement load, a LSTM and FFN model were prepared. In this section, the building and data preparation of these models are discussed whereas the results are shown in the next section.

Input and output data structure for the periodic loading is different from the data of the creep simulations. Firstly, a displacement sinusoidal load at different frequencies was applied to the rubber at the kinematic coupling master node and the reaction forces were recorded at the same location. The simulation duration for all cases was three whole periods. Input to the DL model is the loading displacement and output is the measured reaction forces. Thus, this time both input and output series are magnitudes that increase by time. To make sure that the recorded output is in steady state and no mode excitement is involved in the result which could lead to a dynamic noise, only the last period of the 3 whole periods were saved as input and output data. First two periods might include undesired dynamic effects especially at the frequencies close to the eigenmodes.
Due to the viscoelastic behavior, loading with higher frequency(higher rate) will give a higher reaction force response due to the stiffer behavior. This means that, the DL models have to

be aware of the concept of the time or frequency dependency in order to differentiate the responses associated to the identical load magnitude. Loading displacement as the only input to the DL model has actually no time info. It is only a sinusoidal time series with 100 points per period as resolution and independent from the frequency considered, it will always have 100 points per period where the minimum and maximum amplitude peaks occur at the same resolution point. To overcome this issue, 3 LSTM models were prepared with different input data structure.
First model has as input only the loading displacement, the second one has an extra time series as input whose variables are the time snapshots. The latter is defined in terms of seconds of the simulation(extracted automatically from ABAQUS output) and it increases from 0 to the last second of each simulation. As each simulation has different frequency for the same load amplitude, the end of the time observation window measured in seconds is different for the different frequencies considered and this is how the DL model becomes aware of the time/frequency dependency. The third LSTM model has also an extra input series additional to the displacement load. As time variable, instead of using seconds, the value of the frequency in terms of Hz is given for each simulation as time variable.
All 3 LSTM models have similar hyper-parameters and data preparation. There are 450 simulation results in total as available data, 360 of them were selected as training data and the rest 90 as test data. LSTM architecture is the same as the one used for the creep model(encoder-decoder with one bidirectional LSTM cell for encoder and decoder separately). This time each LSTM cell has 100 neurons. Each time series in input and output has 101 time points which is one whole period. Activation functions of LSTM nodes is rectified linear unit. Optimization algorithm is selected as Adam and the loss function is mean square error. Epoch number is selected as 800 iterations and batch size is 360 which is the whole training set size. Input and output data scaling between 0 and 1 was performed also for these models.

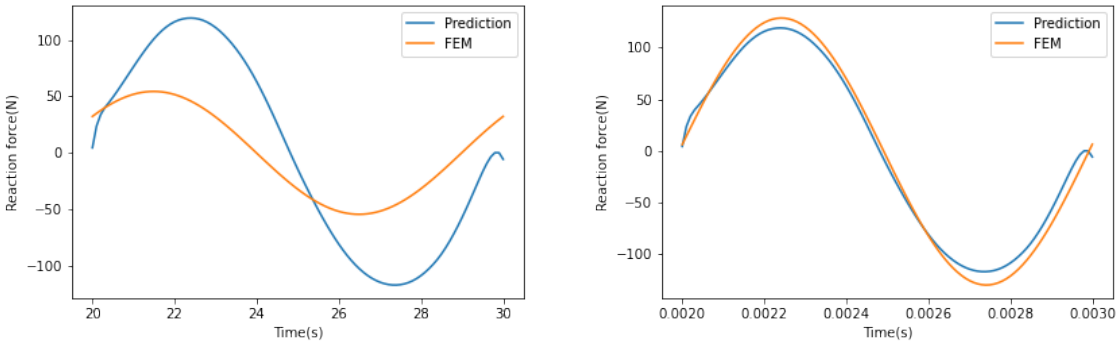
Like these 3 LSTM model variants, also 3 FFN model variants have been prepared analogously to compare the results. FFN models can this time perform a good prediction of results without needing a time series processing model like LSTMs. This is because the time information is also included in the last 2 DL models as extra input variable and now the FFN model can be aware of the time/frequency dependency. This extra time input can enable an input/output mapping for the FFN. Gers et al. denote on their papers that, for most of the cases, a regression problem can only be a autoregression that requires a simple mapping between inputs and outputs [6]. In such cases, LSTM may not utilize its real performance as it is actually designed for way more complicated, mostly future prediction problems, for example natural language processing.

FFN models can not process time series data as a whole series. For that reason, all the series were split into single data lines like the FFN model for the creep loading prediction. Train and test data splits are respectively 80% and 20%. Model consists of three hidden layer with 50 neurons each. Input-output scaling, optimizer and hidden layer activation functions are the same as for the LSTM model. Epoch is set as 850 iterations and the batch size is 360.

4.2.2 Deep learning model results for the periodic load case

Training of the LSTM model only with displacement time series as input took 8 hours and the other models with extra time or frequency series input took around 10 hours for 800 epochs. On the other hand, FFN models took approximately 5 minutes to train for each model. This big time difference is already expected as FFN neurons just do a straightforward mapping instead of holding extra data from the past increments or processing them with gates like

LSTM cells do. Like in the creep model, output results were scaled back to their original values from the 0-1 range. In fig. 4.6, the comparison of FEM results and LSTM model with only displacement as input can be seen for the cases with a displacement load of 0.057 mm at 0.1 Hz and 1000 Hz. Accuracy of the predictions for all the other load frequencies except 0.1 Hz are pretty much similar to the 1000 Hz case. This prediction accuracy dependency on frequencies is the same for other load magnitudes as well. For that reason, only the results of 0.1 Hz and 1000 Hz for the same load magnitude are illustrated in the figures.



(a) Result for 0.057 mm load at 0.1 Hz

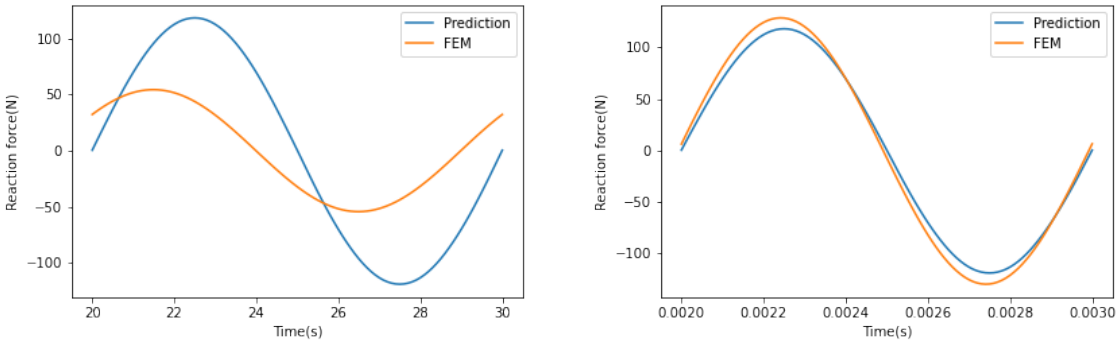
(b) Result for 0.057 mm load at 1000 Hz

Figure 4.6: LSTM and FEM results for two different load cases when LSTM is given only displacement as input.

As seen in fig. 4.6, the accuracy of the prediction for 1000 Hz is quite good, even though no time data is provided to the DL model. However, the accuracy is very bad for 0.1 Hz load case prediction. The reason for this behavior is that, the displacement time series which is the input load to these DL models have the same pattern for each force magnitude, they do not contain any info about frequency or time. So the displacement series, which is 1 whole period with 100 points resolution is exactly the same for 0.1 Hz, 1000 Hz and all the other frequencies. However, the output which is the reaction force time series looks different both in terms of reaction force magnitudes(different reaction to different frequency due to rate dependency) and time shift from the input series(0.1 Hz has more shifted points in the output series as 100 points resolution is pretty fine for it, whereas 1000 Hz does not have so many shifted points). As the reaction forces of 0.1 Hz are way lower than the reaction force results of all the other frequencies, the DL model prediction assumes that the others are true and 0.1 Hz is an outlier. For this reason, an extra time information as input variable is needed for the DL model.

The results of the FFN models for the same simulations are shown in fig. 4.7. As it can be seen in fig. 4.7, FFN model also exhibits a similar prediction to the LSTM model. They both have accuracy problems when it comes to predicting the result of the 0.1 Hz loading case. But for the other load frequencies, the results from both these DL models are quite accurate even though there are some differences between maximum reaction force magnitudes of the DL model and FEM.

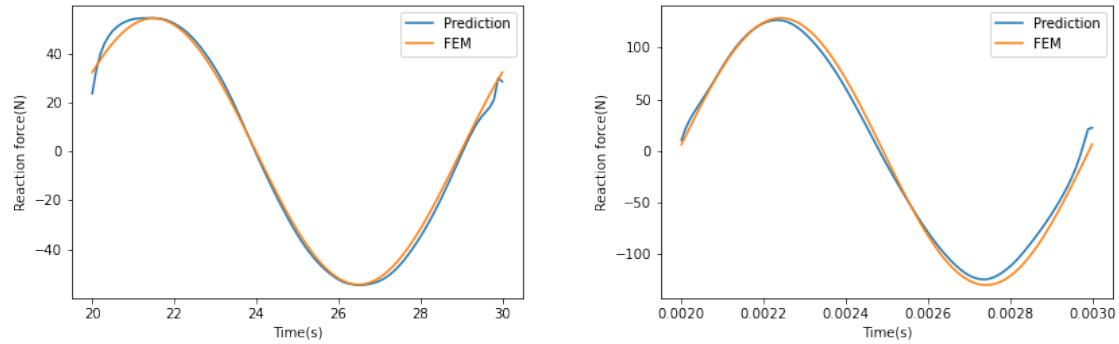
To solve the problem of the wrong prediction for 0.1 Hz, and also to have a better practiced DL model, an extra time series containing the time information of the simulated data in terms of seconds has been supplied to the LSTM model. The results of this model for the same load cases shown above can be seen in fig. 4.8. Similarly, also an FFN model was prepared to predict the same output and its results can be seen in fig. 4.9.
It is clear that, the inclusion of the time data as a separate series has led to a much more



(a) Result for 0.057 mm load at 0.1 Hz

(b) Result for 0.057 mm load at 1000 Hz

Figure 4.7: FFN and FEM results for two different load cases when FFN is given only displacement as input.



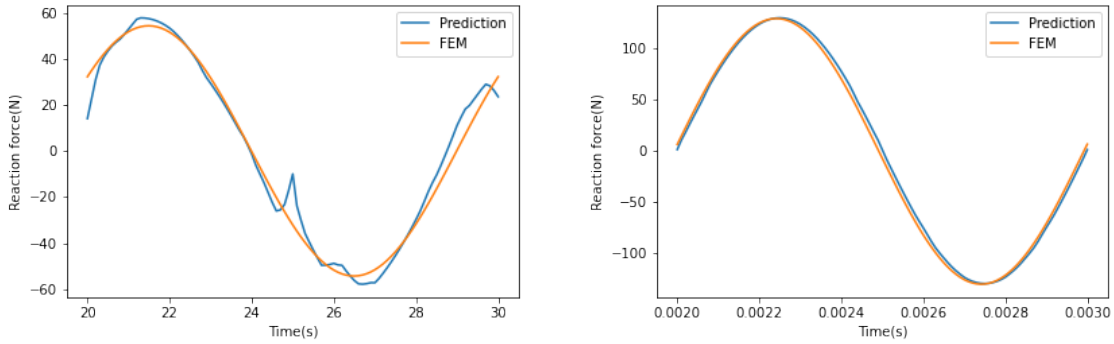
(a) Result for 0.057 mm load at 0.1 Hz

(b) Result for 0.057 mm load at 1000 Hz

Figure 4.8: LSTM and FEM results for two different load cases when LSTM is given both displacement and time as input.

accurate prediction for the loads with 0.1 Hz frequency. Furthermore, also the accuracy for the case with 1000 Hz has been slightly improved.

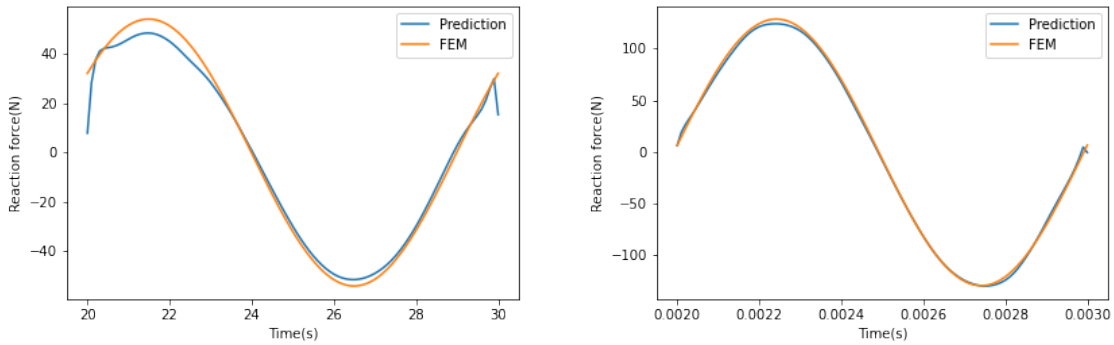
And as the last DL model variation, a new series including the load frequency information in it was introduced as input to the DL models instead of the time data. The input is simply a series including a constant value(applied load frequency) for each 101 time points. This method was applied for both LSTM and FFN models. Results for the LSTM model are shown in fig. 4.10, whereas fig. 4.11 illustrates the results of the FFN model.
As seen in fig. 4.10 and fig. 4.11, the predictions for the 1000 Hz loading cases are very accurate and this is actually valid for all test data predictions except 0.1 Hz, even though not all of them are shown in this thesis. For the load cases with 0.1 Hz, the accuracy of the LSTM prediction is slightly worse compared to the LSTM model with time snapshots as extra input series, even though it is still much better than the prediction of the LSTM model with only displacement data as input. And when it comes to the FFN model, the accuracy of the 0.1 Hz prediction is low as the input series with the frequency includes only the constant frequency values for all of the 101 time points in a series. As the FFN model processes the data as a single input/output time snapshot instead of considering the whole time series, it becomes confusing for the FFN, when the same frequency amplitude as input gives output reaction forces with different magnitudes at different time snapshots and this reduces its accuracy.



(a) Result for 0.057 mm load at 0.1 Hz

(b) Result for 0.057 mm load at 1000 Hz

Figure 4.9: FFN and FEM results for two different load cases when FFN is given both displacement and time as input.



(a) Result for 0.057 mm load at 0.1 Hz

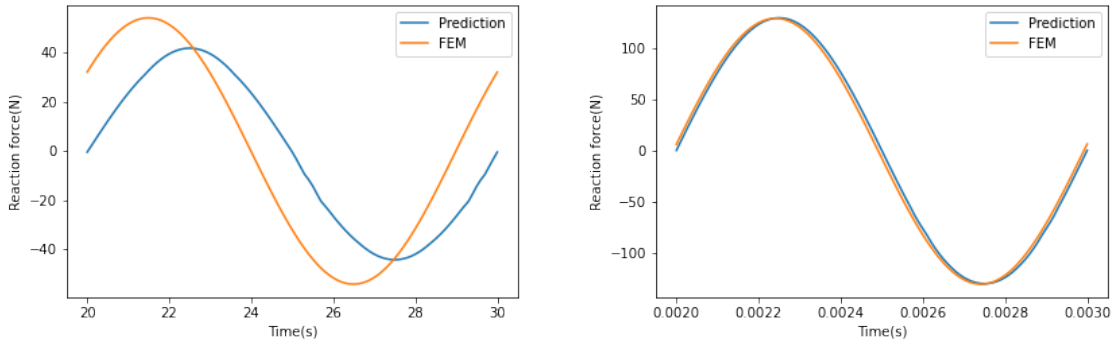
(b) Result for 0.057 mm load at 1000 Hz

Figure 4.10: LSTM and FEM results for two different load cases when LSTM is given both displacement and frequencies as input.

From all the DL models and their results above, it can be inferred that the major part of the periodic load result prediction is actually a auto-regression problem that does not need a very complicated LSTM model. Especially considering that training LSTM takes 8-10 hours whereas FFN only needs 5 minutes and mostly reaching similar prediction accuracy, FFN might be a better choice for such a modeling approach. As an exception, LSTM is mostly better than FFN model for the 0.1 Hz loading case and giving the time series input to the model can enable the 0.1 Hz compete with the LSTM model too in terms of accuracy.
In the overall evaluation of the periodic loading case, a relatively simple FFN model can be adequate for this type of mapping problems as LSTMs are mainly designed for more complex time series relations. However, it has to be noted that, both FFNs and LSTMs perform well for all loading cases, when the time information is provided as an extra input variable, otherwise the predictions of the DL may not be successful at some frequencies.

4.3 Programming aspects of the deep learning models

Tensorflow/Keras is utilized to build the DL models. Keras is an API for deep learning written in Python. Keras has the advantage of being very flexible and it enables the experimentation of new model/implementation ideas in a very fast way to obtain results. Keras runs on the



(a) Result for 0.057 mm load at 0.1 Hz

(b) Result for 0.057 mm load at 1000 Hz

Figure 4.11: FFN and FEM results for two different load cases when FFN is given both displacement and frequencies as input.

machine learning platform Tensorflow, which is an open source platform used for low level tensor operations and differentiable programming on central process unit(CPU) or graphics processing unit(GPU). All the building layers, neurons and other properties of a DL model are included in Keras and the computation of the model is performed on Tensorflow.
For the LSTM models, the encoder-decoder architecture was built with different layers in the model. First a bidirectional LSTM layer is inserted via the ”Sequential” module of Keras. This module contains every building layer and other DL related attributes like compilation/fitting of a model. After the bidirectional layer, a repeating vector layer is added to obtain the output of the first LSTM layer as a sequence data. Without the repeating vector, the output would be a single value instead of a sequence. After the repeating vector, the second bidirectional LSTM layer(decoder) is added. This layer takes the output of the repeated vector as input and gives the output as a sequence. As the last layer, a time distributed dense layer is added after the second LSTM. It is important for this last output layer to be time distributed in order to get the final output as a time series instead of a singular output value. LSTM architecture is the same for both creep and periodic loading cases. Only difference is the dimension adjustments of the neural layers depending on if the input has only one variable or it has also a second variable which is the time information.
FFN models have a relatively straightforward architecture. Similar to the LSTM models, ”Sequential” module of Keras is utilized for FFN models as well. Only dense layers with neurons are implemented in FFN models that are input, hidden and output layers respectively.

Chapter 5

Conclusion and future work

5.1 Conclusion

A deep learning approach was utilized in this thesis to create a surrogate model that can replace the finite element model of a rubber component. Rubber materials exhibit hyper-elastic and viscoelastic characteristics. Two loading cases showing this behavior are modeled in ABAQUS which are a creep response to a constant magnitude load application and reaction to a periodic displacement loading. Obtained simulation results were utilized as data input to the DL models.
FFN and LSTM models were built to predict the FEM results of both the load cases. Even though the cases in this thesis seem to be a time series prediction problem, which require the usage of RNN architecture, a FFN model was also experimented for the predictions. This is because FFN models take much shorter time to train as their computational complexity is less than RNNs and they require a smaller memory usage, hence it is an advised practice to try them before RNNs [6]. A time series prediction problem that tries to figure out the relationship between an input and output series can actually be an auto-regression problem(output is a mapping of input). In that case, a FFN could be used to solve this auto-regression without needing RNNs. Among the RNNs, LSTM(and specifically bidirectional LSTMs due to using the future and past time steps as input) were chosen to overcome the exploding or vanishing gradient problems of regular RNNs.
Prediction of the creep simulation results with LSTM model gave very accurate results for different training data sets. This model has been tried for a case where the test data load magnitudes are in the range of the train data magnitudes and the predictions were very accurate. As RNNs are known to lose accuracy while predicting a test data whose magnitude is out of the train data range, another model was tried for this setup and LSTM still gave high accuracy results for the test data whose load magnitudes were out of the bonds of the training data. This shows the good extrapolation capability of LSTM cells. On the other hand, FFN model, as expected, could not predict the creep behavior. The reason for this is that the input time series consist of constant force magnitude, however, the output(creep response) is a changing displacement amplitude. This impairs the mapping ability of the FFN model.
While predicting the response of the periodic loading, 3 different model variations were developed both for LSTM and FFN architectures. First variation had simply only the loading displacement time series as input without any time data provided and it gives the reaction forces as output series. Second variation included a time series consisting of the simulation seconds in ascending order as an extra input and the third variation had an extra input series that contains the magnitude of the frequency at which the force was applied. From the results of the DL model predictions, it was observed that both FFN and LSTM models had a good accuracy compared to the FEM results for the loads applied at every frequency except 0.1 Hz.

However, when it comes to the load cases at 0.1 Hz, second and third variation models of LSTM and only the second variation model for FFN architectures could reach a high accuracy. The reason for the less accuracy of the other DL models is that, 0.1 Hz loading response of the rubber is much softer than the other loading frequencies which means that, the output reaction force series magnitudes are way lower than the other frequency responses. When no time data is given to the DL model as extra input to the loading time series, the DL models can not figure out this big difference in the response magnitudes, even though the input loading series is the same for all frequencies. For this reason, it is important to introduce the time data as an input as well. Training of the LSTM models take between 8 and 10 hours whereas FFN models require only 5 minutes. As can be seen in the results, the periodic loading prediction problem is actually an auto-regression problem when the time data is provided to the model. For such a case, a relatively straightforward FFN model can be more logical to use compared to a complicated LSTM model.

5.2 Future work

The accuracy of the LSTM models can be increased, especially when one needs to use it to predict the results of much more complicated FEM problems. There is a big room for DL model design trial iterations. Changing the layer and neuron number, increasing the epoch to a very high number are the simplest attempts, even though an increase in these parameters will not lead to a better results after a point [8]. This is because the accuracy is limited by the available data(DL models always improve with more data). The optimizer already gets closest to the global minimum of the loss function after enough epochs with the available data. More epochs or more neuron layers in this case either will not increase accuracy or lead to overfitting of the model to the training data. Overfitting the training data reduces the accuracy on the test data instead of improving. One can also try to combine the LSTM cells with convolutional neural networks or build a different LSTM design instead of a encoder-decoder type for a higher prediction accuracy. DL models give users a high flexibility and many design combinations to try for finding the best results and the best DL model is always dependent on the type of the problem in the first place.
The developed DL models in this thesis can also be implemented in a commercial FEM solver software or a MBS software as a user-built material routine. In this way, the big computational load of a complex rubber material can be eliminated by replacing it with a DL model which only takes seconds to predict the response of the rubber material without needing to solve the constitutive equations of the material and the FEM.

Bibliography

[1] Adamczak, S. and Bochnia, J. “Estimating the Approximation Uncertainity for Digital Materials Subjected to Stress Relaxation Tests”. In: *Metrology and Measurement Systems* (2016), pp. 545–553.

[2] Cho, K., Merriënboer, B. van, Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. “Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proc. of 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1724–1734.

[3] Choi, H.-S., An, J., Kim, J.-G., Jung, J.-Y., Choi, J., Orzechowski, G., Mikkola, A., and Choi, J. “Data-driven simulation for general purpose multibody dynamics using deep neural networks”. In: (Sept. 2019).

[4] *CS 230 - Deep Learning*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[5] Doh, J., Lee, J., and Kim, S. “Reliability Assesment on the Degradation Properties of Polymers under Operating Temperature and Vibration Conditions”. In: *Journal of Automobile Engineering* (2017), p. 6. DOI: 10.1177/0954407017735263.

[6] Gers, F., Eck, D., and Schmidhuber, J. “Applying LSTM to Time Series Predictable through Time-Window Approaches”. In: *International Conference on Artificial Neural Networks* (2001), pp. 669–676. DOI: 10.1007/3-540-44668-0_93.

[7] Ghavamian, F. and Simone, A. “Accelerating Multiscale Finite Element Simulations of History-Dependent Materials Using A Recurrent Neural Network”. In: *Comput. Methods Appl. Mech. Engrg. 357* (2019).

[8] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[9] Hochreiter, S. and Schmidhuber, J. “Long Short-Term Memory”. In: *Neural Comput. 9* (1997), pp. 1735–1780.

[10] Karlsson, F. and Persson, A. “Modelling Non-linear Dynamics of Rubber Bushings - Parameter Identification and Validation”. Master’s thesis. Lund University, 2003.

[11] Kelly, P. *Solid Mechanics Part I: An Introduction to Solid Mechanics*. Mechanics Lecture Notes. University of Auckland, 2013.

[12] Kriesel, D. *A Brief Introduction to Neural Networks*. 2007.

[13] MacKnight, W. and Montgomery, T. *Introduction to Polymer Viscoelasticity*. Wiley-Interscience, 2005.

[14] Namini, S., Tavakoli, N., and Namin, A. “The Performance of LSTM and BiLSTM in Forecasting Time Series”. In: *2019 IEEE International Conference on Big Data(Big Data)*. 2019. ISBN: 978-1-7281-0858-2.

[15] Nielsen, M. *Neural Networks and Deep Learning*. Determination Press, 2015.

[16] Orangi, S. “Time-Dependant Behavior of Ramming Paste Used in Hall-Heroult Cell: Characterization and Constitutive Law”. PhD thesis. Laval University, 2014.

[17] Smith, M. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp, 2009.

[18] Varsamopoulos, S., Bertels, K., and Almudever, C. "Designing Neural Network Based Decoders for Surface Codes". In: (Nov. 2018).

[19] Vincent, J. *Structural Biomaterials*. Princeton University Press, 2012.

Disclaimer

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Garching, February 25, 2022

(Signature)