A Fast Neural Network-Based Computational Framework for the Prediction of Skin Contraction.

M. W. Schaaphok

Delft University of Technology

TU Delft

A Fast Neural Network-Based Computational Framework for the Prediction of Skin Contraction.

by

M. W. Schaaphok

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday November 5th at 10:00 AM.

Student number:	4355822	
Project duration:	February 10, 2020 – November 5, 2020	
Thesis committee:	Prof. Dr. ir. F. J. Vermolen,	TU Delft/UHasselt, supervisor
	Dr. ir. M. B. van Gijzen,	TU Delft
	Dr. ir. F. H. van der Meulen,	TU Delft

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

TU Delft

Preface

This thesis is written as part of completing the master program Applied Mathematics at the Delft University of Technology. It is part of a larger research project, supported by the Dutch Burns Foundation, which is focused on developing a mathematics-based application for predicting skin contraction for burn patients. Specifically, over the last nine months, I have focused on the application of neural networks to provide a computationally cheap alternative for the expensive predictions of the mathematical models.

I was drawn to this topic by one of my personal experiences. In 2019, one of my friends sustained a burn injury that required medical care in one of the burn centers in the Netherlands. My friend had been lucky and did not suffer from long-term effects, but the accident did make me aware of the consequences of burn injuries. Having this experience motivated me to work on this topic and to find methods to increase the applicability of mathematical models in burn care. In the end, I believe that this technology can lead to an application that will help medical staff and will improve the life of severe burn victims.

Moreover, I enjoyed working on this topic as it challenged me to discover the world of machine learning. Over the course of this research, I have learned a lot about machine learning, its challenges, pitfalls, and the endless possibilities.

I am grateful to all the wonderful people surrounding me, who supported me during this project and who gave me advice on many different topics. In particular, I want to give special thanks to my supervisor, Fred Vermolen, for his guidance during this research, his optimism, and for our enjoyable Skype meetings. Furthermore, I want to thank Ginger Egberts for allowing me to use her implementation of the one-dimensional model, for helping me navigate through the web of skin parameters and for her feedback. Likewise, I am grateful to Antonio Barion for providing me with his implementation of the two-dimensional model. I also want to thank Kees Lemmens for giving me access to the computer cluster for a few weeks to generate the datasets, which made it possible to generate data for the two-dimensional model as well. Lastly, I want to thank my boyfriend for making working at home more fun, for the lovely lunch walks and for his unwavering support.

M. W. Schaaphok
Delft, October 2020

Abstract

Burn injuries occur daily and can have severe physical and mental effects both in short and long term, such as disabilities due to severe skin contraction. Even though the mortality rate has decreased over the years, the need for a higher quality of life after severe burns remains. Decreasing the probability of a severe contraction is essential for increasing the quality of life. Mathematical models have been developed to predict skin contraction over time. However, the computations are time-expensive and not suitable for applications that require many simulations, for example, when considering input uncertainty for patient-based predictions.

To that end, the application of neural network surrogates is studied to accelerate the computations of a morphoelastic numerical model for the prediction of skin contraction. Two datasets are generated from a one- and two-dimensional model respectively and are used to train the neural networks. It is shown that a feedforward neural network can accurately learn the nonlinear mapping between the input parameters and the outputs of the considered morphoelastic models. The trained neural network provides fast and accurate predictions on skin contraction and strain energy. Furthermore, a first step is taken towards a hybrid model where a neural network is applied as a surrogate for computationally expensive time-stepping in the numerical model.

The added value of fast neural network surrogates is demonstrated in two clinical case studies. It is shown that the surrogate can be used to perform input parameter studies by comparing an age study with the morphoelastic model with the age study using the neural network. The neural network can reproduce the age study with high accuracy in just a fraction of the time. Secondly, a concept application is designed to demonstrate patient-based predictions using Monte-Carlo simulations to cope with input parameter uncertainty. The application provides predictions for skin contraction and the strain energy, based on the age of the patient and the wound size.

Contents

Acronyms

ANN Artificial Neural Network.

aRRMSE average Relative Root Mean Squared Error.

CNN Convolutional Neural Network.

GPU Graphics Processing Unit.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

MLP Multilayer Perceptron.

MSE Mean Squared Error.

NN Neural Network.

PCA Principal Components Analysis.

ReLU Rectified Linear Unit.

RMSE Root Mean Squared Error.

RNN Recurrent Neural Network.

RSAW Relative Surface Area Wound.

SGD Stochastic Gradient Descent.

SVD Singular Value Decomposition.

1 Introduction

1.1. Introduction & Motivation

Burn injuries are common injuries that can cause severe physical and mental effects both in the short and long term. The WHO estimates around 180 000 deaths a year from burns and nearly 11 million burn injuries worldwide that require medical care [1]. More specifically, in the Netherlands every year 92 000 people need to be treated for burn injuries of whom 9000 require direct first aid and 900 need hospitalization [2]. Burn injuries are associated with substantial morbidity and decrease the quality of life long term [43]. Even though prevention campaigns have reduced the number of burn injuries the last couple of years, burn injuries still occur daily [69, 88]. Over the years better treatments have improved the terms of survival following a burn injury, but the effects on a patient's life after a burn injury remain severe. Therefore finding methods to improve the quality of life after severe burns is essential. To achieve this, more developments on burn treatments are needed to further improve the outcome of burns, acutely, and in the long term [43].

One of the main long-term effects caused by severe burns is reduced mobility in the burned area due to contraction and hypertrophic scarring. During wound contraction, a biomechanical interaction causes the edges of the wound to pull inwards, reducing the area and deforming the wound. If long-term reduced mobility occurs, it is commonly named a contracture. The severity of the contracture depends on wound size, location on the body, and the extent of the contraction. Contraction causes much pain and discomfort to the patient and can result in life-long disabilities affecting the patient's future.

Improving burn injury treatments is an active research topic, with developments in the used dressings [41], the application of skin grafts [5, 38], cell therapies [56], methods to improve self-care of burn patients [9], and more. However, it is a time expensive and difficult task to obtain good experimental data for new treatments, where ethical questions should also be taken into account. Combining data from different studies and experiments can lead to new discoveries, though these studies are often limited due to differences in data collection, environments and documented variables [90].

Mathematical modeling can provide an alternative method to research the effects of individual elements as it allows the user to experiment more freely and more focused on specific effects and relations. The application of mathematical models has gained much interest in the past decades and has become more adopted in the industry and our everyday life. Mathematical models that capture the physical and biological processes can increase our understanding of the underlying biological systems and can assist in medical procedures such as diagnosis and radiation therapy [79]. In the case of burn injuries, these detailed models can give insight into which elements of the healing process have a major influence on the contraction. Over the years various mathematical models have been developed to predict the behavior of wound healing and contraction, such as Koppenol [50], Murphy et al. [63], Olsen et al. [68], Sherrat and Murray [85], Tranquillo and Murray [92], and Egberts et al. [23]. For overviews of the developed mathematical models for burn injuries we refer to Buganza Tepole and Kull [16], Sheratt and Dallon [84] and Vermolen [94].

Furthermore, there is a growing interest in personalized healthcare, an approach focused on tools for delivering patient-centered, predictive care [89]. Here mathematical models can provide a solution,

since the parameters can be tuned relatively easily to achieve patient-based predictions. The patient-based predictions can assist medical staff in making the optimal treatment choices for each patient. The predictions can also increase the knowledge and manage the expectations of the patients about future contraction and pain, which might also improve their self-care. It should be noted that there is often much uncertainty regarding the values of parameters for individual patients. To cope with these uncertainties and to predict the probability of success (no contracture) or failure (contracture) it is necessary to perform Monte-Carlo type simulations, requiring many model-based predictions for one patient [98] .

The requirement of many model-based predictions can be problematic, considering that a downside of high-dimensional mathematical models is that they are often time-consuming. Modeling biological processes results in very large systems as biological systems are complex and often consist of many different interconnected layers. The studies by Noble [67] and Reed [77] provide interesting views of mathematical biology and the challenges of modeling biological phenomena. Noble predicts that this century mathematical biology will become the most intensive science from a computational point of view. Hence applications of the models are and will be limited by the available computational resources. To increase the applicability of the mathematical models, researching methods to reduce or avoid these computer-intensive computations is essential. In the context of predicting contractures for burn wounds, it is not feasible for medical staff to wait days or weeks for predictions on the probability of a contracture for a single patient. For a solution we look to Artificial Neural Networks (ANNs), which are known for their ability to approximate complex relations with relatively high accuracy and within short evaluation time after sufficient training [27].

The application of neural networks and deep learning in the medical society has increased over the past few years especially for identification purposes. Computer vision has been used to identify the degree of skin burns [42] and to identify tumors in MRI scans [60]. Neural networks have been also been applied to identify diseases, such as COVID-19, in blood samples [15]. There is an increased interest in the combination of mathematical physics-based models and machine/deep learning to utilize the advantages of both techniques. The review by Alber et al [6] gives an overview of the opportunities and challenges of combining these techniques for applications in the biomechanical and social fields. One of the opportunities they name is to use neural network surrogates to accelerate slow numerical models. In this approach, a neural network is trained to predict the nonlinear mapping between inputs and outputs of a slow mathematical model. There exists the possibility to provide predictions with high accuracy and fast evaluation, which increases the applicability of the mathematical model for personalized healthcare and treatment research. In this research computationally cheap methods, using surrogate neural networks, are studied for skin contraction models developed by Egberts et al. [23] and Barion [12]. The application of the surrogate neural networks is studied in two clinical cases.

1.2. Related work

In this section, related research on the application of surrogate neural networks for accelerating computational expensive models is discussed. The integration of neural networks and numerical models can be done in multiple ways. For example, they can be used as a surrogate or proxy for a complete model based on simulations by the model, or as a replacement for one (repeated/iterative) time-expensive computation within the model. The integration of numerical models and neural networks can also be used for the approximation of difficult parameters estimations/parameterizations [17, 28], for model reduction [57, 97] or they can be used as a replacement of numerical methods and learn to solve systems of differential equations [36, 46, 55, 75]. Note that this is not an extensive list and other methods for combining neural networks and numerical models have been developed as well, for instance by [11, 18, 37]. In the next paragraphs, a small overview is given of the work that has been done relating to a surrogate approach for a complete numerical model and for an iterative time-expensive computation.

The first approach employs a neural network as a surrogate for a numerical model, this neural network is trained to learn the nonlinear mapping between the inputs and outputs of the numerical model. This surrogate approach is applied by C.Yang et al. [20], where a Convolutional Neural Network (CNN) is used to accelerate the approximation of the stress-strain curve for materials, using Principal Components Analysis (PCA) to reduce the dimension. Following a similar approach, Wang et al. [95] considered a Long Short-Term Memory (LSTM) neural network to accelerate mechanical models used for studying the dynamics of biological systems. The trained surrogate allowed them to scan through

a large parameter space which would have been unfeasible with the original model. Furthermore, Navratil et al. [65] have shown that a neural network can outperform other, non-intelligent, acceleration techniques on both acceleration and accuracy. A neural network embedded Monte Carlo approach is studied by Zou et al. [98], where the neural networks functions as a surrogate for a numerical water quality model. They showed that this approach has the potential for efficient analysis of the input uncertainty. The neural network surrogate approach has also been applied for environmental numerical models by Krasnopolsky and Chevallier [52], who illustrated four different applications of simple neural networks in environmental numerical models, and for urban wastewater systems by Ráduly et al. [81], where a mechanistic model for a wastewater treatment plant was replaced by a neural network. Moreover, the use of neural networks has been studied for high complexity systems in oil reservoir modeling [59, 65] and computational fluid dynamics [8, 24, 35].

We can conclude from these researches that neural networks can be used as fast surrogates for mathematical models, although the accuracy of the trained networks varies dependent on the difficulty of the task and available data.

The second approach where a neural network is used as a surrogate for a computationally expensive step has also been studied. In Grzeszezuk et al. [33] the control and emulation of multiple physics-based graphics models were replaced by a neural network to accelerate the simulations and reduce computational demand. The authors trained a neural network to approximate the function ϕ: $x(t + \Delta t) = \phi(x(t))$, which could then be applied repeatedly to accelerate the computations. They also showed that the neural networks can take larger time steps compared to the original models without serious loss of accuracy. Meister et al. [58] explore the use of deep learning to accelerate the time-integration of Total Lagrangian Explicit Dynamics with neural networks that are generic enough to handle various geometries, motion and materials. Furthermore, Krasnopolsky and Fox-Rabinovitz [53] have shown conceptual and practical possibilities to develop hybrid models that combine machine learning components for time-consuming physical components and deterministic components. They showed that the hybrid approach produced very similar results in significantly less time.

It is evident that the combination of neural networks and mathematical-physics based models is an active research topic with many variations. Therefore, it is interesting to study to what extent this combination can accelerate the finite element models for skin contraction and improve their applicability.

1.3. Approach & Contribution

The objective of this research is to find a computationally cheaper method using neural networks to reproduce expensive numerical models for skin contraction after severe burns. The new method should increase the applicability of the skin contraction model for parameter studies and healthcare by reducing computation time and maintaining accuracy. In this section, the approach to solving this problem is discussed. The common approach using a numerical model for the prediction of biological processes is as follows. First, a mathematical description of the biological process is developed in terms of (partial) differential equations and characteristics. The solutions to the derived system of equations are approximated using numerical methods, such as finite difference, finite volume, or the finite element method. The numerical approximations are then used as the final predictions for the behavior of the biological process. Solving the systems numerically is often time-expensive and the computations need to be performed for each new set of inputs for which a prediction is required. These slow predictions take place ‘online’, i.e. at the time that the user requires the predictions.

A neural network surrogate has the advantage that most of the work can be done ‘offline’, i.e. before the user needs to compute a prediction. First, the numerical model is used to construct a dataset of inputs and corresponding predictions. The construction of this dataset is very time-consuming due to the number of simulations necessary for the training. This construction can fortunately be done offline and can benefit greatly from parallel programming on servers. The size of the constructed dataset is dependent on the problem at hand and should be manageable. The dataset is divided into a training set and a test set. The training set is used to teach the network how to predict the outputs from the inputs and the test set is used to validate the performance of the network. The time needed to train the network is highly dependent on the type of network, the type of data, and the amount of data. However, as both training and validation of the neural network can be performed offline, training time is not a major concern. Once the neural network has been trained to give accurate predictions, it can be used to compute new predictions. Note that the accuracy of the neural network is with respect to the

predictions by the numerical model and not with respect to the true solutions. Due to their structure, neural networks can provide online predictions instantaneously. Hence, instead of the slow online predictions computed by the numerical model, the surrogate neural network can be used to obtain fast online (approximations of the) predictions. A downside of the approach is that the use of neural network surrogate loses the physical interpretation of the mathematical model. The approach is visualized in Figure 1.1.

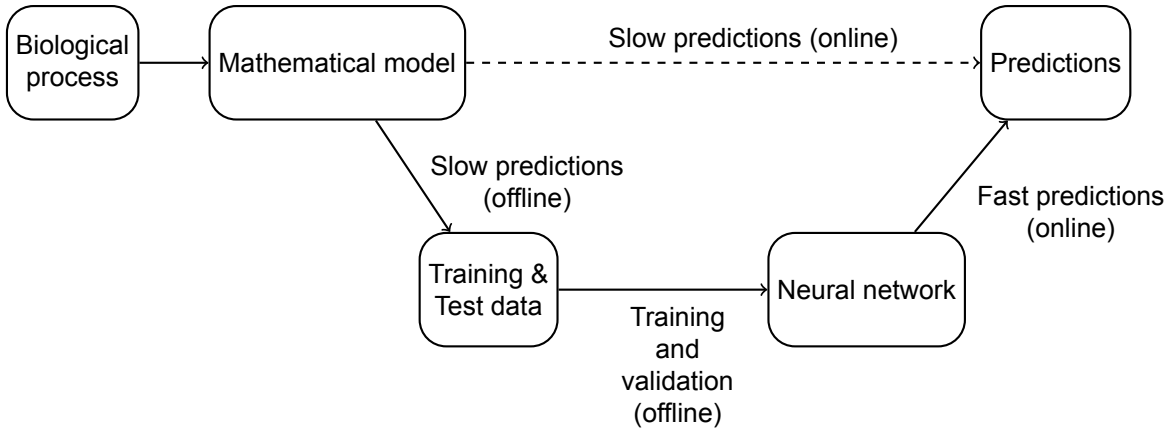


Figure 1.1: Approach to accelerate the predictions of the biological healing of burn wounds. A neural network surrogate is used to provide accurate approximations of the predictions made by a slow mathematical numerical model.

In this research, multiple methods are studied to apply the surrogate approach for fast and accurate predictions of skin contraction models. The study is built on the mathematical models for skin contraction developed by Egberts et al. [23] and Barion [12]. First, the neural network is used as a full surrogate for the numerical model. In this approach the network is trained to predict the desired outputs, such as contraction over time, directly from the input parameters. Secondly, a hybrid approach is developed, where the neural network is used as a surrogate only for the most computationally expensive step. In the hybrid approach the network is trained to predict all variables at the next time step, based on their value at the current time step. The applicability of the neural network surrogate is tested in two clinical case studies. The first case study is a parameter study, where the neural network is used to find the influence of the patient's age on the wound contraction. The study with the neural network is compared to the study with the finite element model for validation. In the second case study a concept application for medical staff is designed using the neural network to perform fast Monte Carlo simulations. The clinical case studies provide clear examples of how the neural network surrogates can benefit research on the effects of parameters and how they can be used for personalized healthcare.

1.4. Outline

The outline of the thesis is as follows. In Chapter 2 the mathematical framework of the morphoelastic model is discussed. The one-dimensional model is discussed in detail, whereas for the two-dimensional model only the main differences are highlighted. Chapter 3 describes the applied methods, including the basic methods for machine learning, the methods for data generation and data processing, and the applied performance measures. Chapter 4 discusses the results for a full neural network surrogate for the one- and two-dimensional numerical model. Surrogates for both the prediction of relative surface area and strain energy are trained and the effect of principal component transforms and training set size is investigated. The study for the two-dimensional model includes a surrogate for the prediction of the wound edge movement as well. Chapter 5 discusses the second approach for the one-dimensional model where a surrogate is trained for the most expensive computational step only, i.e. the finite element prediction of one time step. Chapter 6 discusses two clinical case studies. First, a parameter study is performed using the neural network surrogate to investigate the influence of age on skin contraction. The neural network-based study is compared to the study with the morphoelastic model for validation. Secondly, this chapter introduces a concept application for medical staff providing patient-based predictions on the contractures. Chapter 7 provides the conclusion and the discussion of the study and gives recommendations for future research.

2

Morphoelastic Model for Burn Injuries

In this research, neural network surrogates are studied for both the one- and two-dimensional version of a morphoelastic model, developed and implemented by Egberts et al. [22] and Barion [12]. respectively. The main part of the study is built upon simulations of the one-dimensional model. Therefore, this chapter explains the mathematical framework of the one-dimensional morphoelastic model in detail. For the two-dimensional model only the main differences are highlighted as the two models are based on the same mathematical principles.

2.1. One-dimensional morphoelastic model for burn injuries

The mathematical framework of this model is based on the general morphoelastic model for burn wound contraction, developed by Koppenol [50]. The derivation and implementation of the one-dimensional model are done by Egberts et al. [23]. Firstly, the mathematical framework of the model is discussed, considering the differential equations, constitutive relations, and initial and boundary conditions. Next, the computations of relevant output variables as the relative surface area of the wound and the strain energy in the wound are explained and the applied numerical methods are mentioned.

2.1.1. Mathematical framework

The model considers four biological constituents and three mechanical components. The four biological constituents are the fibroblasts (N), the myofibroblasts (M), a generic signaling molecule (c), and collagen (ρ). The considered mechanical components include displacement of the dermal layer (u), the displacement velocity of the dermal layer (v) and the effective strain in the dermal layer (ϵ). The model is described by Equations (2.1), (2.2), and (2.3).

$$\frac{\partial z_i}{\partial t} + \frac{\partial (z_i v)}{\partial x} = -\frac{\partial J_i}{\partial x} + R_i, \tag{2.1}$$

$$\rho_t \left(\frac{\partial v}{\partial t} + 2v \frac{\partial v}{\partial x} \right) = \frac{\partial \sigma}{\partial x} + f, \tag{2.2}$$

$$\frac{\partial \epsilon}{\partial t} + v \frac{\partial \epsilon}{\partial x} + (\epsilon - 1) \frac{\partial v}{\partial x} = -G. \tag{2.3}$$

Equation (2.1) is the conservation of the cell density/concentration for each of the constituents. Here z_i represents each of the constituents, $i \in \{N, M, \rho, \varepsilon\}$, J_i is the flux of the constituent i per unit area and R_i represents the chemical kinetics of constituent i. Equation (2.2) is the conservation equation of linear momentum, where ρ_t is the total mass density of the dermal layer, σ is the stress tensor, and f is the body force. Lastly, Equation (2.3) shows the evolution equation of the strain in the skin, where G is the growth tensor.

The constituents

First, the fluxes and chemical kinetics in Equation (2.1) are described for each of the constituents. In the remainder of this chapter, z_i is replaced by i, so z_N is represented by N, z_ρ by ρ etc. The fluxes J_i in the equation for the (myo)fibroblasts describe both their random movement and their directed movement due to chemotaxis. We define $F = M + N$, D_F as the random diffusion parameter and χ_F as the chemotaxis, such that the fluxes J_N and J_M are given by Equations (2.4) and (2.5), respectively.

$$J_N = -D_F F \frac{\partial N}{\partial x} + \chi_F N \frac{\partial c}{\partial x}, \tag{2.4}$$

$$J_M = -D_F F \frac{\partial M}{\partial x} + \chi_F M \frac{\partial c}{\partial x}. \tag{2.5}$$

Equation (2.1) also includes a reaction term that describes the proliferation of the cells. It includes the differentiation of myofibroblasts into fibroblasts and the removal of these cells due to apoptosis (cell death). The chemical kinetics for the (myo)fibroblasts R_N and R_M are given by Equations (2.6) and (2.7), respectively.

$$R_N = r_F \left[1 + \frac{r_F^{\max} c}{a_c^I + c}\right][1 - \kappa_F F] N^{1+q} - k_F c N - \delta_N N, \tag{2.6}$$

$$R_M = r_F \left[\frac{[1 + r_F^{\max}] c}{a_c^I + c}\right][1 - \kappa_F F] M^{1+q} - k_F c N - \delta_M M. \tag{2.7}$$

The main difference between both equations is that myofibroblasts only proliferate in the presence of the signaling molecules, while fibroblasts also proliferate without the signaling molecules. The parameter r_F is the cell division rate, $r_F^{\max}$ the maximum factor of the cell division rate enhancement due to the signaling molecules, a_c^I the concentration of signaling molecule that causes half-maximum enhancement of the cell division rate, κ_F represents the reduction in the cell division rate due to crowding, k_F is the signaling molecule-dependent cell differentiation rate of the fibroblasts into myofibroblasts, and δ_N, δ_M are the apoptosis rates of the fibroblasts and myofibroblasts, respectively.

Next, the flux and the reaction term for the signaling molecules are considered. The flux J_c only includes the random movement of the signaling molecules and is given in Equation (2.8), where D_c the random diffusion coefficient.

$$J_c = -D_c \frac{\partial c}{\partial x}. \tag{2.8}$$

The proliferation of the signaling molecules, and hence the reaction term, depends on both the local density of the (myo)fibroblasts that secrete and consume the signaling molecules and a generic metalloproteinase (MMP) that removes the signaling molecules through a proteolytic breakdown. The concentration of MMP is proportional to the cell density of the (myo)fibroblasts and the concentration of collagen and signaling molecules, and is given in Equation (2.9).

$$g(N, M, c, \rho) = \frac{[N + \eta^{II} M]\rho}{1 + a_c^{III} c}. \tag{2.9}$$

Note that Equation (2.9) is based on the assumption that the MMP concentration follows instantaneously from the presence of (myo)fibroblasts, collagen, and the generic growth factor. Here η^{II} is the ratio of myofibroblasts to fibroblasts in the secretion rate of the MMPs and a_c^{III} is the concentration of the signaling molecule that causes inhibition of the secretion of the generic MMP. The proliferation of the signaling molecule R_c is then given by Equation (2.10).

$$R_c = k_c \left[\frac{c}{a_c^{II} + c}\right]\left[N + \eta^I M\right] - \delta_c g(N, M, c, \rho) c. \tag{2.10}$$

Here k_c is the maximum net secretion rate of the signaling molecule, η^I is the ratio of myofibroblasts to fibroblasts in the maximum secretion rate of the signaling molecule, a_c^{II} is the concentration of the signaling molecule that causes half-maximum net secretion rate of the signaling molecule, and δ_c is the proteolytic breakdown rate of the signaling molecules.

For collagen we assume no active transport, and therefore the flux is equal to zero, i.e. $J_\rho = 0$. The proliferation of collagen by the (myo)fibroblasts is described in the reaction term R_ρ. The secretion rate of collagen k_ρ is enhanced by the presence of signaling molecules, where the maximum factor of secretion rate enhancement due to the signaling molecules is given by $k_\rho^{\max}$. Furthermore, a_c^{IV} is the concentration of signaling molecules that causes the half-maximum enhancement of the secretion rate. The proteolytic breakdown of collagen is analogous to the breakdown of signaling molecules and δ_ρ gives the degradation rate of collagen. Equation (2.11) combines this information and gives the reaction term R_ρ.

$$R_\rho = k_\rho \left[1 + \left[\frac{k_\rho^{\max} c}{a_c^{IV} + c}\right]\right] \left[N_\eta^I M\right] - \delta_\rho g(N, M, c, \rho)\rho. \tag{2.11}$$

The mechanical components

Next, the mechanical components of the model are considered. The Cauchy stress tensor σ, the body force f and the growth tensor G are given in Equation 2.12, where μ is the viscosity and $E\sqrt{\rho}$ the Young's modulus.

$$\sigma = \mu \frac{\partial v}{\partial x} + E\sqrt{\rho}\epsilon, \qquad f = \frac{\partial \psi}{\partial x}, \qquad G = \alpha\epsilon, \qquad \alpha \in \mathbb{R}, \tag{2.12}$$

The total stress ψ is generated by the myofibroblasts and is given by Equation (2.13).

$$\psi = \left[\frac{\xi M \rho}{R^2 + \rho^2}\right]. \tag{2.13}$$

Here ξ is the generated stress per unit cell density and the inverse of the unit collagen concentration and R is a constant. Equations (2.2) and (2.3) model the conservation of mass and linear momentum. A tensor based approach, commonly used in the context of tissue growth, is used to incorporate a permanent contraction. The growth factor in this case models the contraction of the tissue and is assumed to be proportional to the product of the cell density of (myo)fibroblasts and a function of the collagen density. Moreover, it is assumed that the contraction term depends on the product of the concentration of MMPs, the concentration of the signaling molecules, and the reciprocal of the collagen density. The growth rate $\alpha\epsilon$ is given in Equation (2.14), where ζ denotes the rate of morphoelastic change.

$$\alpha\epsilon = \zeta \left\{\frac{[N + \eta^{II} M]c}{1 + a_c^{III} c}\right\} \epsilon. \tag{2.14}$$

Initial and boundary conditions

The domain of the computation is defined by $\Omega = [-L_1, L_1]$, with boundary $\partial\Omega = \{-L_1, L_1\}$. The initial wound area is defined by the subdomain $\Omega^w = [-L_2, L_2]$, where $L_2 < L_1$ and the boundary of the wound is defined by $\partial\Omega^w = \{-L_2, L_2\}$. The steepness of the boundary, which also defines the slopes of the components, is defined by s. The dimension of x is in cm and t is in days. The initial cell density for the fibroblasts is given by Equation (2.15), where $\bar{N}$ is the initial fibroblast cell density in healthy dermal tissue and $\tilde{N}$ is the initial fibroblast cell density in the wounded area.

$$N(x,0) = \begin{cases} \bar{N} & \text{if } x \le -L_2, \\ \frac{\bar{N}+\tilde{N}}{2} + \frac{\bar{N}-\tilde{N}}{2}\sin(\frac{\pi}{s}(x + \frac{1}{2}s)) & \text{if } -L_2 \le x \le -L_2 + s, \\ \tilde{N} & \text{if } -L_2 + s \le x \le L_2 - s, \\ \frac{\bar{N}+\tilde{N}}{2} + \frac{\bar{N}-\tilde{N}}{2}\sin(\frac{\pi}{s}(x + \frac{1}{2}s)) & \text{if } L_2 - s \le x \le L_2, \\ \bar{N} & \text{if } x \ge L_2, \end{cases} \tag{2.15}$$

For the initial concentration of the signaling molecules, the function in Equation (2.16) is used, where $\bar{c}$ is the concentration in healthy dermal tissue and $\tilde{c}$ is the concentration in the wounded area.

$$c(x,0) = \begin{cases} \bar{c} & \text{if } x \le -L_2, \\ \frac{\bar{c}+\tilde{c}}{2} + \frac{\bar{c}-\tilde{c}}{2}\sin(\frac{\pi}{s}(x + L_2 - \frac{1}{2}s)) & \text{if } -L_2 \le x \le -L_2 + s, \\ \tilde{c} & \text{if } -L_2 + s \le x \le L_2 - s, \\ \frac{\bar{c}+\tilde{c}}{2} + \frac{\bar{c}-\tilde{c}}{2}\sin(\frac{\pi}{s}(x + 1\frac{1}{2}s - L_2)) & \text{if } L_2 - s \le x \le L_2, \\ \bar{c} & \text{if } x \ge L_2. \end{cases} \tag{2.16}$$

The initial concentration of the concentration of collagen is defined by Equation (2.17), where $\bar{\rho}$ is the concentration in healthy dermal tissue and $\tilde{\rho}$ is the concentration in the wounded area.

$$\rho(x,0) = \begin{cases} \bar{\rho} & \text{if } x \leq -L_2, \\ \frac{\bar{\rho}+\tilde{\rho}}{2} + \frac{\bar{\rho}-\tilde{\rho}}{2}\sin(\frac{\pi}{s}(x+\frac{1}{2}s)) & \text{if } -L_2 \leq x \leq -L_2+s, \\ \tilde{\rho} & \text{if } -L_2+s \leq x \leq L_2-s, \\ \frac{\bar{\rho}+\tilde{\rho}}{2} + \frac{\bar{\rho}-\tilde{\rho}}{2}\sin(\frac{\pi}{s}(x+\frac{1}{2}s)) & \text{if } L_2-s \leq x \leq L_2, \\ \bar{\rho} & \text{if } x \geq L_2. \end{cases} \tag{2.17}$$

Furthermore, it is assumed that initially no myofibroblast cells are present, and that the displacement of the dermal layer, the displacement velocity and the initial strain are zero. These initial conditions are given in Equation (2.18).

$$M(x,0) = \bar{M} = 0, \quad u(x,0) = 0, \qquad v(x,0) = 0, \qquad \epsilon(x,0) = 0, \quad \forall x \in \Omega_{x,0}. \tag{2.18}$$

The boundary conditions for all $x \in \partial\Omega_{x,t}$ and $t \geq 0$ are given by Equation (2.19).

$$N(x,t) = \bar{N}, \qquad M(x,t) = \bar{M} = 0, \qquad c(x,t) = \bar{c} = 0, \qquad v(x,t) = 0. \tag{2.19}$$

The boundary condition for the displacement velocity stems from the assumption that the boundary of the computation is sufficiently far away from the boundary of the wounded area, and hence remains zero at all times. There is no boundary condition on the concentration of collagen, the displacement of the dermal layer, and the effective strain.

2.1.2. Relative surface area wound

From the mechanical model part, the displacement-field over the domain is obtained, and from the displacement, the relative wound area is determined. The Relative Surface Area Wound (RSAW) is important as it provides information about skin contraction. The surface area at time t is derived by adding the displacement of the wound edge at time t to the initial wound size. The relative surface area is computed using Equation (2.20).

$$\text{RSAW}(t) = \frac{L_2 + u(t, x_b)}{L_2}, \tag{2.20}$$

where L_2 is the initial length of the wound and x_b is the grid point at the edge of the wound.

The relative surface area provides information about the maximum contraction during healing, the number of days until maximum contraction, the final contraction, and the number of days until final contraction, which is shown in Figure 2.1. The minimum of the relative surface area predicts the maximum contraction, and the asymptotic value determines the final contraction. This is valuable information as it describes the area of the scar over time and hence provides interpretable information about the probability of a contracture.

2.1.3. Strain energy

The contraction of wounds and scars causes stress and strain in the skin, and it is imaginable that the stress on the skin can cause pain or discomfort to the patient. Although to our knowledge it has not been thoroughly studied, it possible that the total amount of strain energy at a certain moment is related to the discomfort a patient experiences. If this is the case, the strain energy could prove to be a valuable tool for research as well as for medical staff.

Assuming linear elasticity, the strain energy can be calculated from the Young's Modulus and the effective strain using Equation 2.21.

$$E_{\text{strain}}(x,t) = \frac{1}{2} Y \varepsilon(x,t)^2. \tag{2.21}$$

Here it needs to be taken into account that the elasticity of the skin is dependent on the amount of collagen, which changes during the healing process, and hence the Young's Modulus is dependent on the collagen density $\rho(x,t)$:

$$Y = E\sqrt{\rho(x,t)}. \tag{2.22}$$

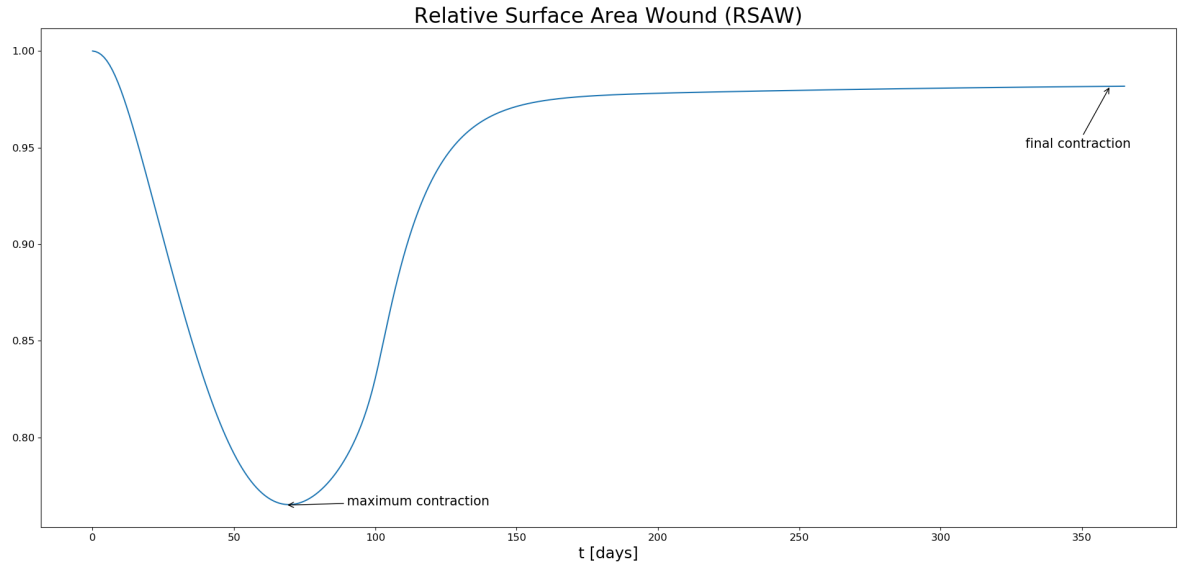


Figure 2.1: Example typical RSAW distribution over time with maximum and final contraction highlighted.

The effective strain $\varepsilon(x,t)$ is dependent on both space and time. As we are interested in the total strain energy that the patient experiences at a certain time, the strain energy is integrated over the domain. Due to the symmetry of the domain, we can integrate over half the domain and multiply by two. Therefore, the strain energy can be computed by Equation (2.23).

$$E_{\text{strain}}(t) = 2\int_0^L \frac{1}{2} E\sqrt{\rho(x,t)}\varepsilon(x,t)^2 dx = \int_0^L E\sqrt{\rho(x,t)}\varepsilon(x,t)^2 dx. \tag{2.23}$$

2.1.4. Numerical methods

The system of differential equations is solved using the finite element method with linear basis functions. For the time integration, the backward Euler method is applied, using a monolithic approach with inner Picard iterations. Further derivations of the numerical methods are omitted as they are not essential for this study and can be found in [22]. The numerical model is implemented in Matlab. As the effective strain and the collagen density are given at discrete points in space, the strain energy integral is approximated numerically using the Simpsons rule.

2.2. Two-dimensional morphoelastic model for burn injuries

The two-dimensional morphoelastic model is based on the same mathematical framework as the one-dimensional model with a few differences. To prevent repetition, only the main differences in the mathematical framework are discussed. For a detailed derivation of the two-dimensional model we refer to the work of Barion [12].

2.2.1. Differences mathematical framework

The general conservation equations for mass and linear momentum are equal to the generalizations of Equations (2.1) and (2.2). The first main difference is the evolution equation of the effective strain which is shown in Equation (2.24) for two dimensions. Here L is displacement velocity gradient tensor, $L = \nabla v$, and it should nog be confused with the length L in the one-dimensional model.

$$\frac{D\epsilon}{Dt} + \epsilon\text{skw}(L) - \text{skw}(L)\epsilon + [\text{tr}(\epsilon) - 1]\text{sym}(L) = -G. \tag{2.24}$$

The second main difference is the mathematical description of the relation between the Cauchy stress tensor, the effective strains and displacement velocities, which is Equation (2.12) in one dimension. In two dimensions, the Poisson effect must be taken into account and both shear and bulk viscosity must be considered. The relation for the two-dimensionsal model is shown in Equation (2.25).

$$\sigma = \mu_1 \text{sym}(L) + \mu_2 \left[\text{tr}(\text{sym}(L))I\right] + \left[\frac{E\sqrt{\rho}}{1+\nu}\right]\left\{\epsilon + \text{tr}(\epsilon)\left[\frac{\nu}{1-2\nu}\right]I\right\}. \tag{2.25}$$

Here μ_1 is the shear viscosity, μ_2 is the bulk viscosity, ν is the Poisson ratio, $E\sqrt{\rho}$ is the Young's modulus, and I is the second-order identity tensor.

Furthermore, the domain of computation and the initial conditions are slightly different. A square domain with area of 64 cm^2, i.e. $\Omega = [-4,4]\times[-4,4]$ cm is considered, with a centered squared wound of initial area of 4 cm^2, i.e. $\Omega^w = [-1,1]\times[-1,1]$ cm. Due to symmetry, only a quarter of the domain is considered resulting in the domain $\Omega = [0,4]\times[0,4]$, with the wound domain $\Omega^w = [0,1]\times[0,1]$.

The initial conditions for c, N and ρ change as well. The initial condition for the myofibroblasts M remains $M = 0$ across the complete domain. The initial conditions are constructed using an indicator function I_{Ω^w} for the wound area. The initial conditions with the indicator function are shown in Equations (2.26) - (2.28).

$$N(x,0) = \bar{N} - (\bar{N} - \tilde{N})I_{\Omega^w}, \tag{2.26}$$

$$c(x,0) = \tilde{c}I_{\Omega^w}, \tag{2.27}$$

$$\rho(x,0) = \bar{\rho} - (\bar{\rho} - \tilde{\rho})I_{\Omega^w}. \tag{2.28}$$

We use the following function, based on the $\tanh$, to approximate the indicator function:

$$I_{\Omega^w} = (0.5 + 0.5\tanh(k(x+1)))\cdot(0.5 - 0.5\tanh(k(x-1))), \quad k \geq 0. \tag{2.29}$$

This function is exactly one on the wound domain and zero outside for $k \to \infty$ and can easily be extended to the defined two-dimensional domain. An advantage of the function is that the gradient through the wound edge can be varied by the adapting the value of k. The value $k = 50$ for the initial conditions.

The boundary conditions for the velocity field are homogeneous Dirichlet conditions and the boundary conditions for the constituents are their equilibrium values in healthy skin, i.e. $M = \bar{M}, N = \bar{N}, c = \bar{c}$. The problem is solved using the finite element method with bilinear elements. More information on the numerical approach can be found in [12].

2.2.2. Relative surface area wound

As explained for the one-dimensional model, the relative surface area of the wound is an important measure that can give information about the maximum and the final contraction. To determine the relative surface area of the wound, the movement of the boundary grid points is derived from the computed velocities. At each day the area of the convex hull, determined by the boundary grid points, is computed. This area is divided by the original area to obtain the relative surface area of the wound over time.

The relative area of the wound or scar is a measure of contraction, though, it does not indicate any localized contractions of the scarred skin. It might be of interest to define localized contractions as well, especially when considering spatially varying input parameters.

2.2.3. Strain energy

In two dimensions, the derivation of the strain energy becomes slightly more complicated due to shear stress. In general, the energy integral is given by Equation (2.24) [82].

$$E_{\text{strain}}(t) = \int_\Omega U(x,y,t)\, d\Omega = \int_\Omega 1/2\left[\epsilon_{xx}\sigma_{xx} + 2\epsilon_{xy}\sigma_{xy} + \epsilon_{yy}\sigma_{yy}\right] d\Omega. \tag{2.30}$$

Here U is the strain energy at a given point and Ω is the domain. The stress and strain components are related through Hooke's Law for plain stress, which can be found in Equation (2.31).

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{bmatrix} = \frac{1}{1-\nu^2}\begin{bmatrix}E_x & \nu E_x & 0\\ \nu E_y & E_y & 0\\ 0 & 0 & G(1-\nu^2)\end{bmatrix}\begin{bmatrix}\epsilon_{xx}\\ \epsilon_{yy}\\ 2\epsilon_{xy}\end{bmatrix}, \tag{2.31}$$

Here E_x and E_y are the Young's modulus in x and y direction respectively, and G is the shear modulus of elasticity.

It is assumed that the elasticity of the skin is not dependent on direction, but on the concentration of collagen in the skin and hence $E_x = E_y = E = \tilde{E}\sqrt{\rho(x,y,t)}$. Furthermore, the shear modulus of elasticity can be written as a function of Poisson ratio and Young's modulus for isotropic material, which is shown in Equation (2.32) [91].

$$G = \frac{\tilde{E}\sqrt{\rho}}{2(1+\nu)}. \tag{2.32}$$

Combining Equations (2.30), (2.31), and (2.32 gives the following expression for the total strain energy in the domain:

$$E_{\text{strain}}(t) = \int_\Omega 1/2\left[\epsilon_{xx}\sigma_{xx} + 2\epsilon_{xy}\sigma_{xy} + \epsilon_{yy}\sigma_{yy}\right] d\Omega \tag{2.33}$$

$$= \int_\Omega \frac{1}{2}\frac{1}{1-\nu^2}\left[\epsilon_{xx}(\epsilon_{xx}E + \nu E\epsilon_{yy}) + 2\epsilon_{xy}\frac{E}{2(1+\nu)}\epsilon_{xy} + \epsilon_{yy}(\nu E\epsilon_{xx} + E\epsilon_{yy})\right] d\Omega \tag{2.34}$$

$$= \int_\Omega \frac{1}{2(1-\nu^2)}\left[\tilde{E}\sqrt{\rho}\left[\epsilon_{xx}^2 + 2\nu\epsilon_{xx}\epsilon_{yy} + \frac{1}{1+\nu}\epsilon_{xy}^2 + \epsilon_{yy}^2\right]\right] d\Omega. \tag{2.35}$$

Since a square domain is used, and the problem is symmetric along the x- and y-axis, the integral can be reduced to a quarter of the domain. The strain energy can therefore be computed by Equation (2.36), where L is half of the length of each side of the domain.

$$E_{\text{strain}}(t) = \int_0^L \int_0^L \frac{2}{(1-\nu^2)}\left[\tilde{E}\sqrt{\rho}\left[\epsilon_{xx}^2 + 2\nu\epsilon_{xx}\epsilon_{yy} + \frac{1}{1+\nu}\epsilon_{xy}^2 + \epsilon_{yy}^2\right]\right] dxdy. \tag{2.36}$$

3

Machine Learning Methods

This chapter covers the applied methods in this research. The first part of this chapter gives a basic understanding of machine learning and is largely based on the book 'Deep learning' by Goodfellow et al. [31]. In Section 3.1, the process of designing a neural network and its main challenges are discussed. Following this outline, Sections 3.2 and 3.3 discuss the theory of feedforward and convolutional neural networks and neural network training. Section 3.4 discusses the importance of and difference between validation and testing. Furthermore, the methods regarding data generation and data processing are discussed in Sections 3.5 and 3.6, respectively. Lastly, Section 3.7 explains the applied performance measures.

3.1. Designing a neural network

In short, a machine learning model is an algorithm that performs a certain task, e.g. prediction of a value or label, based on known input data. The main difference with a common numerical model is that it is not specified how to predict the output from the input. Instead, the model contains a neural network which consists of connected layers of so-called neurons. Each neuron has learnable parameters that are trained to give a prediction based on the inputs. The training is performed by feeding the network a dataset containing inputs and the corresponding desired outputs. An optimization algorithm then updates the values of the learnable parameters such that they minimize a predefined loss function. After training, validation and testing are necessary to prove if the model has been trained correctly. In this section, the process of designing a neural network and its challenges are discussed.

3.1.1. Design process

Designing a well-functioning neural network can be challenging because of its large dependence on the data and the objective. There is no universally best type of network or architecture, no best optimization algorithm, and no best parameter combination. This means that often much of the design comes down to trial and error and tuning parameters to find a combination that works well for the given task and dataset. To guide this process and to prevent losing track of what has already been done, it is important to follow a structured design process and to document the performed trials and experiments. The design process of a neural network is explained below and is visualized in Figure 3.1.

1. **Task definition:** First, it is important to define the task or objective of the neural network. The task can, for example, be the prediction of values or the classification of images. Defining the objective also includes choosing a method to determine the performance of the network. In classification tasks, this can be the percentage of correct predictions. It is advised to have the performance measure return a single number, which makes it easier to compare different networks. Defining the desired performance in advance can provide a guideline through the process.

2. **Data processing:** For the dataset, it is assumed that the generated dataset fits the requirements for the specified task. It is important that the data has a structure in which for each sample the input and output values are easily accessible. Furthermore, the data needs to be split into a training, validation, and testing set to ensure proper evaluation of the network. This is discussed in more detail in Section 3.4.

   To prepare the data for training, task-specific preprocessing is necessary. Examples are cropping for images, zero-padding for sequences, and scaling for numeric values. Especially the scaling of values is essential for the learning process. Training the weights of a model will be hard if the input values have large variations in magnitude. This is explained in more detail in Section 3.6.

3. **Initialization:** The next step is to choose a starting point: a baseline neural network. The choice of the type of neural network is related to the task and it is advised to choose a network that is common for similar problems. For example, one might choose a Multilayer Perceptron (MLP) for function approximation, a Recurrent Neural Network (RNN) for time-series prediction and a Convolutional Neural Network (CNN) for image recognition. The architecture of the network is not only defined by the type, but also by the number of layers, the number of neurons per layer, the topology, and the activation functions. These are the hyperparameters of the network architecture, i.e. parameters that need to be specified in advance and can not be learned by the neural network. Different classes of networks often have different hyperparameters. Besides the architecture, the training algorithm has hyperparameters as well, such as the optimization algorithm, loss function, learning rate, batch size, number of passes through the dataset (epochs), and regularization. All these values need to be initialized with a certain value. Often the first initialization includes commonly used values for each of the parameters. The hyperparameters are discussed in more detail in Section 3.3.

4. **Training:** Using the chosen parameters and network architecture, the network is trained. During training, the data is fed to the network, which predicts the output based on the input. Next the loss, or error, between the prediction and the target is computed. Based on this loss, a gradient-based optimization algorithm updates the learnable parameters of the neural network, such that it minimizes the loss. The details of the training algorithms are discussed in Section 3.3.

5. **Validation:** The trained model is then validated on the validation set, which contains samples the network has not used during training. Based on these predictions and the targets, the performance of the trained network is computed. The performance on the validation set is the measure that is used for the next step: hyperparameter tuning.

6. **Hyperparameter tuning:** Often, the baseline network does not reach the desired performance. Based on the results of the validation set, the hyperparameters can be changed after which steps 3, 4, and 5 are repeated. The model is reinitialized, retrained, and re-evaluated resulting in a new performance. This process is repeated until a satisfactory performance is reached on the validation set. Note here that proper monitoring of the training process and the worst fits in the validation step can help when tuning the parameters. Tuning the parameters can be done by trial and error and therefore proper documentation of each trial is essential.

   There are hyperparameter optimization packages available as hyperparameter tuning can be seen as optimizing the performance of the validation set with respect to the hyperparameters. Using these optimization packages is often computationally expensive, and the ranges in which to search still need to be chosen. Experimenting with trial and error can give some insight into the behavior of the network and might already lead to a good set of hyperparameters. Tuning the hyperparameters to achieve the desired performance is one of the challenging tasks of neural network design.

7. **Testing:** Lastly, the trained model from the best hyperparameters is chosen and evaluated on the test set to find its final performance. The test set should contain samples independent from both the training and validation set. Networks of different classes, with their optimized hyperparameters, can be compared based on their performance on the test set. When the performance on the test set fulfills the requirements, the design process is finished.

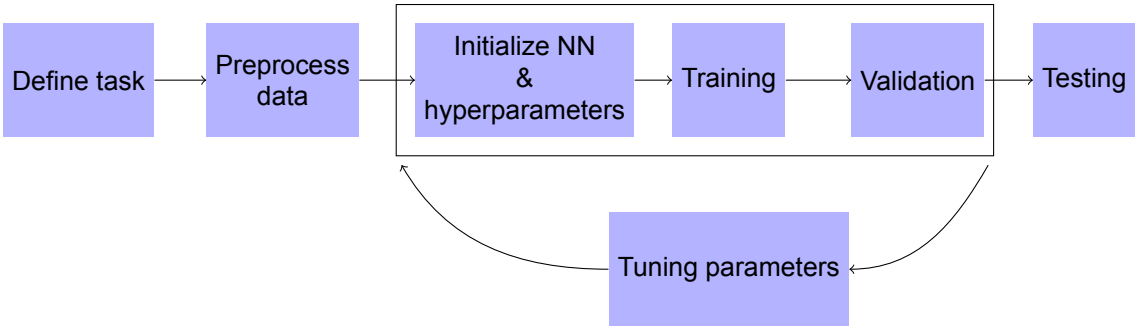


Figure 3.1: Design process of a neural network.

3.1.2. Challenges

As discussed in Section 3.1.1, tuning the hyperparameters is one of the most challenging tasks as there is no universally best combination of parameters, and every problem requires its own approach. Furthermore, one has to deal with the two central challenges in learning algorithms:

1. Making the training error small. When the model is not able to get the training error below a certain threshold, we speak of *underfitting*. The model is not able to capture the relations between input and output in the data set.

2. Reducing the gap between training and validation error. When the training error is much smaller than the validation error, we speak of *overfitting*. In this case the model is not able to cope with data that has not been dealt with in the training set.

In Figure 3.2 the general shape of the training and validation error are shown over the capacity of the model. In the figure the regions of underfitting and overfitting are shown in terms of capacity. The capacity of the network is defined by its configuration, such as the number of layers and the number of nodes in the network. The green area shows a suitable capacity, where the training error is low enough such that the model can capture the relations between input and output and the generalization gap is small enough such that the model can generalize well to unknown examples. It should be noted that in practice both validation and training error show more oscillations due to randomness in training on batches.

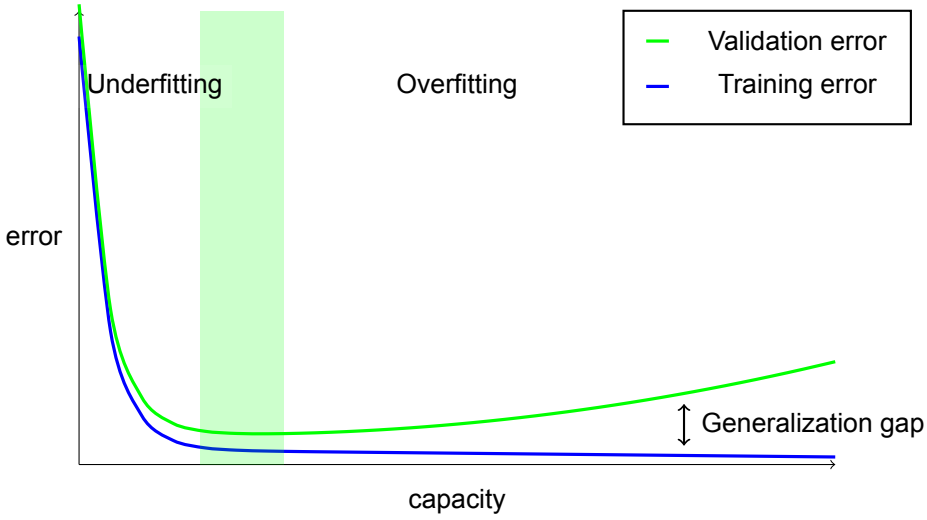


Figure 3.2: General shape of training error and validation error over the capacity of the network. On the left side of the graph the training and testing error are both large, here we speak of underfitting. On the right side the gap between training and validation error increases, which we call overfitting. The green area shows a suitable capacity, the optimal capacity is reached when validation error is minimal. Reproduced from [31].

Underfitting can be recognized by a high training error and can often be resolved by increasing the capacity of the network, such as adding more layers or adding more neurons per layer. As there will always be a slight gap between the validation loss and the training loss, it is more difficult to determine when the model starts overfitting. Overfitting needs to be resolved by decreasing the capacity of the network. One can reduce the capacity by decreasing the number of layers or number of neurons, or by changing the topology. Often, more effective ways to reduce the capacity is to give a preference towards certain solutions, by means of regularization. Regularization techniques are further discussed in Section 3.3.5. Adding more samples to the dataset can also decrease the chance of overfitting.

It must be noted that finding the right capacity does not guarantee a good neural network. Other problems can arise due to the optimization algorithm getting stuck, inconsistent data, wrong data processing, or an implementation error. In order to tackle these challenges, it is important to get a working pipeline as soon as possible with a baseline neural network and to monitor its behavior in terms of training and validation loss. Furthermore, investigating the samples that return the highest error can provide useful insights. Monitoring closely for a simple network increases the chance of finding these types of issues.

In the next sections the neural networks and their training are explained in more detail. We discuss the feedforward neural network architecture and its initialization, the training and optimization algorithms and validation and testing.

3.2. Neural networks

3.2.1. Feedforward neural networks

Feedforward neural networks or Multilayer Perceptrons (MLP) are basic neural networks and many advanced models are derived from an MLP. These networks are used to approximate a mapping $\mathbf{f}$ from inputs $\mathbf{x}$ to outputs $\mathbf{y}$: $\mathbf{y} = \mathbf{f}(\mathbf{x})$. MLP's are proven to be universal function approximators, which means that they are able to approximate a function or mapping arbitrarily well as long as the number of hidden units is large enough [27].

An MLP consists of an input layer with a number of nodes equal to the number of inputs, a number of hidden layers with a specified number of nodes, and an output layer with a number of nodes equal to the desired number of outputs. The feedforward network is actually a mapping $\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{p})$ that tries to learn the values of the parameters $\mathbf{p}$ such that $\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{p}) \approx \mathbf{f}(\mathbf{x})$. In Figure 3.3 the architecture of a simple MLP with an input layer, one hidden layer containing five neurons, and an output layer is shown. The feedforward aspect of the network is clearly illustrated in the figure; the information flows from the input to the hidden layer to the output without any recurrences or feedback.

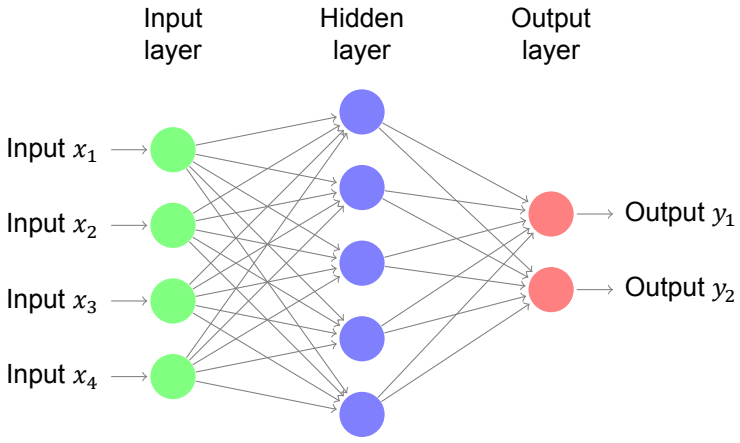


Figure 3.3: Feedforward neural network with four inputs, one hidden layer with five nodes and two outputs.

Before the mathematics is explained in more detail, some terminology of neural networks is discussed. A hidden layer is called *fully-connected* or *dense* if each neuron in a layer is connected to all neurons in the next layer. The number of layers is called the *depth* of the model and hence the name deep learning often refers to networks with multiple hidden layers. The *width* of the network is defined by the dimensionality or number of neurons in the hidden layers.

For the mathematical derivation, the inputs are represented as a vector **x** and the outputs are represented as vector **y**. To be able to approximate both linear and nonlinear mappings, a neural network uses a combination of an affine transformation and a nonlinear activation function. The affine transformation by a hidden unit is defined by $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b$, where **w** is a vector of the weights of each input and b is the bias. If these weights and biases are collected in a matrix W and vector **b** respectively for all neurons in the hidden layer, the transformed vector $\tilde{\mathbf{x}}$ can be computed by Equation (3.1).

$$\tilde{\mathbf{x}} = W^T\mathbf{x} + \mathbf{b}. \tag{3.1}$$

The values are then passed through a fixed nonlinear activation function $g(x)$ to get the hidden values, which is shown in Equation (3.2).

$$\mathbf{h} = g(\tilde{\mathbf{x}}) = g(W_1^T\mathbf{x} + \mathbf{b}_1). \tag{3.2}$$

Examples of frequently used activation functions are the Rectified Linear Unit (ReLU), the sigmoid, or the $\tanh$ function. The activation functions are discussed in more detail in Section 3.2.4. The output layer takes as input the hidden vector **h** and again applies an affine transformation and in some cases a nonlinear activation function. The prediction $\hat{y}$ is then given by Equation (3.3).

$$\hat{\mathbf{y}} = g(W_2^T\mathbf{h} + \mathbf{b}_2) = g(W_2^T g_1(W_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2). \tag{3.3}$$

The weights W_1, W_2, and the biases $\mathbf{b}_1$, $\mathbf{b}_2$, are the *learnable parameters* of the neural network. These learnable parameters are trained such that $\hat{\mathbf{y}} \approx \mathbf{y}$. The training of the learnable parameters is discussed in Section 3.3 and can be seen as solving a least-squares problem using gradient-based optimization.

3.2.2. Convolutional neural networks

CNN's are neural networks specialized in processing structured input data, e.g. data with a grid-like topology such as time-series data or image data. The main building block for the CNN is the convolutional layer, which uses the mathematical convolution operator instead of the standard matrix multiplication. The discrete convolution operator $s(t)$ is given by Equation (3.4).

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{3.4}$$

In neural network application, x is the input and w is the kernel or feature map. We focus the explanation on one-dimensional convolution for an input sequence of length L with a number of channels C_{in}. The kernel is moved over the input sequence with a specified stride. In case the kernel and stride do not fit the length of the sequence, padding can be considered to ensure that all values in the sequence are used. For each output channel c_{out}^j the resulting sequence is given in Equation (3.5), where * is the cross-correlation operator, $\mathbf{b}(c_{out}^j)$ is the bias for the related output channel and $W(c_{out}^j, k)$ are the weights of the kernel with respect to the respective input and output channel.

$$\mathbf{c}_{out}^j = \mathbf{b}(c_{out}^j) + \sum_{k=0}^{C_{in}-1} W(c_{out}^j, k) * \mathbf{x}^k. \tag{3.5}$$

The bias and the weights are the learnable parameters of the neural network.

Similarly to the dense linear layers, the outputs from the convolutional layer are fed through a nonlinear activation function. The convolution layer is often combined with pooling layers and standard dense linear layers to form a complete network. The pooling layers are used to reduce the dimension and extract only the main features and are often applied after the nonlinear activation of the convolutional output. Well-known pooling layers are max pooling and average pooling, where for a small cluster of nodes in one layer the maximum or average value is sent to the next layer. For a more detailed description and visualization of the convolution layer, we refer to [31].

3.2.3. Parameter-sharing neural networks

Neural Networks can easily be extended to multi-target problems, i.e. the prediction of multiple continuous targets from a set of input variables, by adding more output neurons. However, most multi-target problems need to cope with the existence of dependencies between targets, trying to exploit the inter-targets relationships while also being able to map the individual input-target relations. It has been researched whether the architecture of the network can assist in learning both between-target and inter-target dependencies [80].

Furthermore, learning related tasks can add information to the network, which can improve the training of the original task. Multi-task learning can originate from the desire to perform multiple tasks at the same time or from the desire to improve the performance of one specific task by training on auxiliary tasks. In [80] an overview of multi-task learning in deep neural networks is given, discussing both predicting multiple tasks as well as choosing auxiliary tasks. Multi-task learning is often done using either *hard-parameter sharing* or *soft-parameter sharing*. Hard-parameter sharing utilizes architectures where the first few layers are shared among all tasks and the last layers are task-specific. An example of a hard-parameter sharing network can be found in Figure 3.4. The disadvantage of this method is that it is not robust for loosely related tasks. In soft-sharing networks, each task has its own network, but the distance between the parameters of the different networks is regularized to encourage them to be similar for the different tasks.

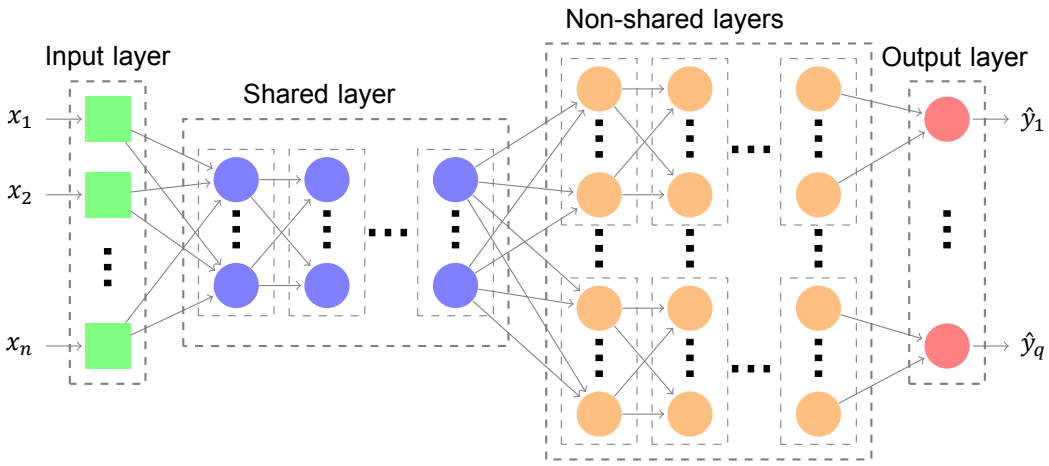


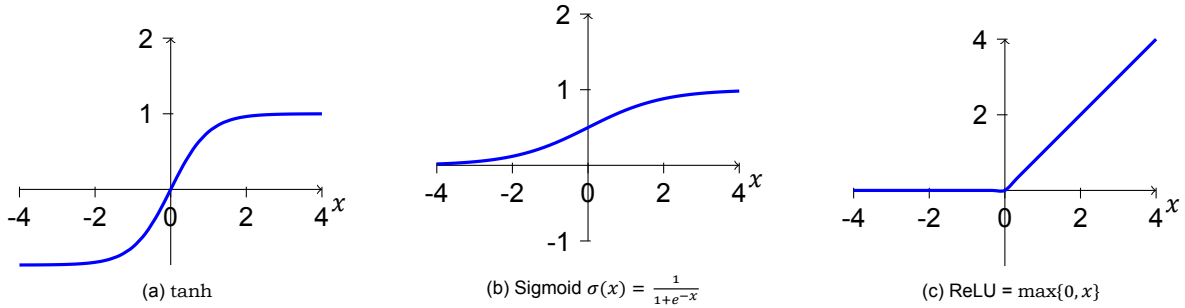
Figure 3.4: Architecture of a hard-parameter sharing based neural network with an input layer, shared hidden layer, non-shared hidden layers and an output layer. Note that each sequence of non-shared layers could also output multiple targets.

3.2.4. Activation functions

As mentioned in Section 3.2.1, the outputs of a hidden layer are often fed through a nonlinear activation function before being sent to the next layer. These nonlinear activations ensure that nonlinear functions can be approximated by the neural network. Without activation functions, the result would always be a linear combination of the inputs, which would greatly restrict the usability of neural networks.

The most commonly used activation functions are the ReLU, the sigmoid function, and the $\tanh$ function, which can be seen in Figure 3.5. The sigmoid function has the advantages that it has a smooth gradient, it normalizes the outputs of the hidden layer and it enables clear predictions. A large downside is that the gradient may vanish if the values of $|x|$ are large which can slow down or even prevent the network from learning. Furthermore, the output values are not zero-centered and the function is more computationally expensive. The $\tanh$ has the same properties as the sigmoid function, except that it is zero-centered, making it more suitable for inputs with strongly negative, neutral, and strongly positive values.

The ReLU function has the advantage that it is computationally efficient as it allows the network to converge quickly. A disadvantage is that it can suffer the 'dying ReLU' problem, which occurs when x approaches zero or becomes negative causing the gradient to become zero. This blocks the learning as backpropagation is not possible anymore. ReLU is a piecewise linear function and for $x > 0$ it is linear.



(a) $\tanh$ (b) Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ (c) ReLU = $\max\{0, x\}$

Figure 3.5: Three common activation functions

A disadvantage is that when $x > 0$, $\forall x$ ReLU loses its ability to introduce nonlinearity and the solution will be linear. In practice, this is unlikely to occur and ReLU is a good choice for introducing nonlinearity.

Small adaptions have been made, such as Leaky ReLU ($\max\{0.1x, x\}$) to prevent the 'dying ReLU' problem, but this does not give consistent results. There is not a best choice for an activation function and different functions should be used to find the best performing one for that specific task.

3.2.5. Initialization

The learnable parameters need to be initialized to compute the first prediction and loss. The values of the initialization are important for the learning process, especially for neural networks with many layers. Stacked hidden layers result in the multiplication of the weight matrices. Due to the number of matrix multiplications, bad initialization of the weights can cause vanishing activation outputs when sent through many layers, which causes vanishing gradients and prevents the network from learning. Keeping the standard deviation of the activation outputs normalized makes it possible to stack many layers without vanishing or exploding gradients.

Initialization methods are developed in such a way that they achieve this standard deviation of approximately one. A commonly used initialization technique is Xavier initialization, introduced as normalized initialization in [30]. With Xavier initialization the weights are drawn from the uniform distribution:

$$W \sim U\left(\frac{-\sqrt{6}}{n_i + n_{i+1}}, \frac{\sqrt{6}}{n_i + n_{i+1}}\right),$$

where n_i is the number of inputs of the ith layer and n_{i+1} the number of outputs of the $i+1$st layer. This causes the activation outputs to have mean 0 and standard deviation 1 and prevents vanishing gradients. It works especially well for activation functions that have outputs between -1 and 1, such as the $\tanh$. For ReLU activation functions, the Kaiming initialization method has been developed [40]. For Kaiming initialization weights are chosen from the standard normal distribution and then multiplied by $\frac{\sqrt{n_i}}{\sqrt{2}}$ and the biases are initialized to zero.

3.3. Training

During training, the training data is passed through the network multiple times to adapt or train the learnable parameters in the neural network. The training consists of three important phases: forward propagation (computing prediction and loss), backward propagation (computing gradients), and optimization (updating learnable parameters). First, the forward propagation algorithm is discussed after which the loss functions are considered in more detail. In Section 3.3.3, the backpropagation algorithm is explained, and in Section 3.3.4 two gradient-based optimization algorithms are discussed. Section 3.3.5 explains commonly used regularization techniques.

3.3.1. Forward propagation

Forward propagation is the computation of the prediction $\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{p})$, the loss $L(\hat{\mathbf{y}}, \mathbf{y})$ and total costs $J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p})$ during the training process. The total costs are a sum of the loss L and a regularization term Γ, which are discussed in Section 3.3.2 and Section 3.3.5 respectively. During forward propagation, the

input values are fed to the first layer, multiplied by its weights and added to its bias, and then passed through a nonlinear activation function before being passed to the next layer. This process is repeated until the output layer is reached giving the prediction. This prediction and the target are used to compute the loss and the total costs. Algorithm 1 shows the process for one single sample $(\mathbf{x}, \mathbf{y})$. We note that in practice the algorithm is performed using sample (mini)batches.

Algorithm 1: Forward propagation. Shows forward propagation for a standard full MLP with ℓ hidden layers and the computation of the total cost incorporating both loss L and a regularization term Γ, discussed in Section 3.3.2 and 3.3.5 respectively. The algorithm only takes into account a single sample $(\mathbf{x}, \mathbf{y})$.

Result: $\hat{\mathbf{y}}, J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p})$
Input: Network depth (ℓ), weights ($\mathbf{W}^{(i)}$), biases ($\mathbf{b}^{(i)}$), input ($\mathbf{x}$), target ($\mathbf{y}$) ;
$\mathbf{h}_0 = \mathbf{x}$;
for *k = 1,...,ℓ* **do**
  $\mathbf{a}^{(k)} = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}$;
  $\mathbf{h}^{(k)} = g(\mathbf{a}^{(k)})$;
end
$\hat{\mathbf{y}} = \mathbf{h}^{(\ell)}$;
$J = L(\hat{\mathbf{y}}, \mathbf{y}) + \lambda\Gamma(\mathbf{p})$;

3.3.2. Loss functions

The training of the parameters is based on the optimization of the loss function. The loss function gives an error metric on the prediction value with respect to the true value or target. Using the target to compute the loss is called supervised training. When the loss function is combined with a regularization term we commonly speak of the total costs J instead of the loss. In this subsection, different loss functions and regularization techniques are discussed. The chosen loss function must fit the structure of the output and the task. Some loss functions are useful for regression problems whereas others are more specialized for classification tasks.

A well known error metric for regression tasks is the Mean Squared Error (MSE). For two general vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ the MSE is defined by Equation (3.6).

$$L^{MSE}(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2. \tag{3.6}$$

The MSE is a standard type loss function for optimization of neural network weights. Other well known loss functions for regression type problems are the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) or L1-loss. These loss functions are shown in Equation (3.7) and (3.8), respectively.

$$L^{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{3.7}$$

$$L^{MAE}(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|. \tag{3.8}$$

The loss function also provides the possibility to incorporate preferences on the solution in the training of the network. For example, if the solution is expected to be smooth a penalty can be added for large variations, or a weighted loss can be used to put more attention on important parts of the solution. Adding information on the solution can improve the networks performance and limit overfitting of the data. However, care must be taken when choosing the weight of the additional condition, such that it does not stop the network from learning from the data. It should also be taken into account that the added condition must be differentiable for the gradient based optimization to work.

To assist the learning of the network and to reduce overfitting, it is also possible to add a regularization term on the learnable parameters to the loss function. The network is then trained by optimizing the total costs instead of only the loss function:

$$J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p}) = L(\hat{\mathbf{y}}, \mathbf{y}) + \gamma\Gamma(\mathbf{p}).$$

Regularization on the learnable parameters is often performed by adding the L1 or L2-norm to the costs to reduce the size of the weights. Reducing the size of the weights can prevent co-adaptation where some weights have large predictive capabilities and others very small. It is important to choose a value for γ that provides a good trade-off between minimizing the loss and minimizing the regularization term. If chosen too large it will prevent learning from the data, whereas if chosen too small it might not perform any regularization. Other well-known regularization techniques for machine learning include early stopping and dropout. However, as these techniques do not use the loss function they are explained later in Section 3.3.5.

3.3.3. Back-propagation

The learning of the parameters is often performed by a gradient-based optimization algorithm that tunes each parameter based on the gradient of the total costs J with respect to that parameter, i.e. $\nabla_{\mathbf{p}} J(\mathbf{p})$. Remember here that the learnable parameters $\mathbf{p}$ are the weights W and biases $\mathbf{b}$. The numerical evaluation of a gradient expression can be computationally expensive and needs to be done many times when training a neural network. The backpropagation algorithm provides an easy and computationally cheap solution, making efficient use of the chain rule for differentiation. Assume $\mathbf{p} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, $J \in \mathbb{R}$ and further it holds that $\mathbf{y} = g(\mathbf{p})$ and $J = J(\mathbf{y})$, which includes no regularization term. The chain rule of differentiation is given by Equation (3.9), where $\frac{\partial \mathbf{y}}{\partial \mathbf{p}}$ denotes the Jacobian matrix of function g.

$$\frac{\partial J}{\partial p_i} = \sum_j \frac{\partial J}{\partial y_j} \frac{y_j}{p_i} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{p}} \right)^T \nabla_{\mathbf{y}} J. \tag{3.9}$$

The idea of obtaining the gradient with respect to parameters $\mathbf{p}$ by multiplying the Jacobian matrix with the gradient, is the backbone of the backpropagation algorithm. For every operation performed, the backprop algorithm constructs the Jacobian matrix and adds it as a node to the computational graph. When arriving at the total costs J, this computational graph can be traversed backwards starting with $\frac{\partial J}{\partial J} = 1$ to obtain the desired gradients. This evaluation can be done with a generic graph evaluation engine and is therefore efficient.

For illustration purposes, we consider a very simple MLP with one input $x \in \mathbb{R}$, one hidden layer consisting of one neuron with weight w, bias b, activation function g, one output $\hat{y}$, and loss function $L(\hat{y}, y)$. Here we assume that no regularization term is added in the total costs, hence there is no direct dependence on $\mathbf{p}$ and $J = L$. In Figure 3.6 the forward propagation and the computational graph of the gradients is illustrated. The backprop algorithm constructs this computational graph for the derivatives. Each of the values can then easily be evaluated as soon as it parents are known.

Most current software packages have a general back-propagation algorithm implemented that can back-propagate through graphs constructed from all general operations like matrix multiplication, exponents, and logarithms. For a full MLP, the backward propagation algorithm, corresponding to the forward propagation in Algorithm 1, is shown in Algorithm 2.

Algorithm 2: Backward propagation for standard MLP with ℓ layers and total cost J incorporating both loss L and a regularization term Γ. Computation starts at the output layers and works back to the first layer, computing first the gradients on the activations $\mathbf{a}(k)$ and then the gradients w.r.t. learnable parameters.

Result: $\mathbf{g}$
$\mathbf{g} = \nabla_{\hat{\mathbf{y}}} J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p}) = \nabla_{\hat{\mathbf{y}}} L(\hat{\mathbf{y}}, \mathbf{y})$;
for $k - l, l - 1, ..., 1$ **do**
  $\mathbf{g} = \nabla_{\mathbf{a}^{(k)}} J = \mathbf{g} \odot g'(\mathbf{a}^{(k)})$;
  $\nabla_{\mathbf{b}^{(k)}} J = \mathbf{g} + \lambda \nabla_{\mathbf{b}^{(k)}} \Gamma(\mathbf{p})$;
  $\nabla_{\mathbf{W}^{(k)}} J = \mathbf{g}\mathbf{h}^{(k-1)T} + \lambda \nabla_{\mathbf{W}^{(k)}} \Gamma(\mathbf{p})$;
  $\mathbf{g} = \nabla_{\mathbf{h}^{(k-1)}} L = \mathbf{W}^{(k)T}\mathbf{g}$;
end

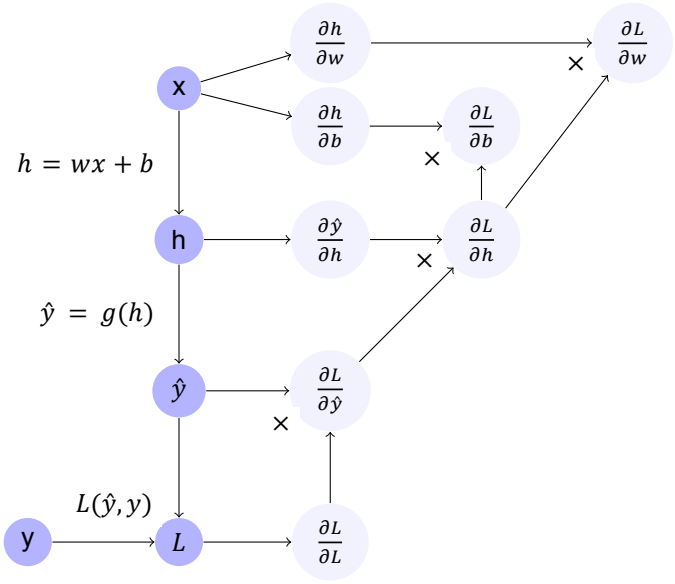


Figure 3.6: Computational graph of $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ for a MLP with one-dimensional input and output, one hidden layer of one neuron, activation function $g(h)$ and loss function $L(\hat{y}, y)$. The blue circles represent the forward propagation through the network and the light blue circles represent the computational graph constructed by the backward propagation.

3.3.4. Optimization algorithms

After forward and backward propagation the gradient of the total costs J with respect to each learnable parameter is known. The computed gradients can directly be applied in a gradient-based optimization method such as Stochastic Gradient Descent (SGD) or the Adam algorithm [49] to update the parameters. Similar to the forward and backward propagation, we consider the unregularized case where $J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p}) = L(\hat{\mathbf{y}}.\mathbf{y})$.

In machine learning, the training is commonly performed using minibatches of samples of the training set. The reason to use these minibatches stems from the properties of the gradient-based optimization algorithms. Considering each sample individual would give large variations in the gradients and would require a very small learning rate to prevent the large variations from interrupting the learning. Using the complete set at once is often not possible or not desired in terms of runtime or memory. It can be shown that the average gradient over a minibatch gives an unbiased estimate of the true gradient when the samples are drawn independently. Note here that as soon as we pass through the same sample another time the samples are not independent anymore and the estimate is no longer unbiased. Another advantage of using minibatches is that each update keeps the same runtime when the size of the training set is increased.

The minibatch SGD or more commonly named stochastic gradient descent performs the standard gradient descent on each minibatch of random samples. The standard gradient descent method is an optimization method that uses the fact that the gradient shows the direction of the largest increase of a function. When considering the function $f(\mathbf{x})$ that we want to minimize, computing the gradient $\mathbf{g} = \nabla_{\mathbf{x}} f(\mathbf{x})$ gives the direction of the largest increase. Since we want to minimize, a step into the opposite direction of the gradient is taken:

$$\mathbf{x} \leftarrow \mathbf{x} - \lambda \mathbf{g} = \mathbf{x} - \lambda \nabla_{\mathbf{x}} f(\mathbf{x}).$$

The size of the step λ, called the learning rate, is an essential parameter for the performance of the algorithm. Choosing the learning rate too large may result in the algorithm jumping over the minimum, while taking it too small may result in a very long run time or never reaching the minimum. Hence the learning rate is one of the important hyperparameters in machine learning. In theory, the learning rate can be a fixed number as the gradient decreases until it is zero, However, in practice with stochastic gradient descent, the learning rate needs to be decaying. The necessity arises from the noise introduced by the random sampling in the gradient estimation which prevents the gradient from vanishing completely. Unfortunately, there are no fixed guidelines on how to choose these parameters. In practice, proper initial values can be found by monitoring the learning curves of the objective function.

The SGD algorithm is shown in Algorithm 3. A disadvantage of SGD is that it can be slow to converge, solutions for this disadvantage can be found in the acceleration technique using the method of momentum. This technique accelerates the convergence at the cost of introducing an additional parameter that needs to be initialized. For more information about SGD with momentum, we refer to [31]. Also, conjugate gradient methods have been used to obtain faster convergence [44],[66].

Algorithm 3: Stochastic gradient descent (SGD) update at training iteration k.

Result: Updated parameters $\mathbf{p}$
Input: Learning rate λ_k, parameters $\mathbf{p}$
while *stopping criterion not met* **do**
  Sample a minibatch of m samples from training set $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_m, \mathbf{y}_m)\}$
  Compute gradient estimate: $\hat{\mathbf{g}} \leftarrow \frac{1}{m} \nabla_{\mathbf{p}} \sum_i J(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$.
  $\mathbf{p} \leftarrow \mathbf{p} - \lambda_k \hat{\mathbf{g}}$.
end

The learning rate is a parameter that is essential for machine learning algorithms, but that is difficult to initialize. To make the initialization easier, methods with adaptive learning rates have been developed. These new, minibatched-based, optimization methods adapt the learning rate of each model parameter individually. Examples are AdaGrad, RMSProp and Adam of which we only discuss Adam in more detail. Adam is derived from adaptive moments and can be considered as a variant on RMSProp. More details about AdaGrad and RMSProp can be found in [31].

Both RMSProp and Adam are effective and popular optimization algorithms for neural networks. Adam has the advantages that momentum is directly incorporated as an estimate of the first-order moment of the gradient and it includes bias corrections to both first- and second-order moments. It can be found in Algorithm 4. The Adam algorithm is considered to be fairly robust to the choice of hyperparameters, which is a preferable property. Often, the suggested defaults of $\lambda = 0.001$, $\rho_1 = 0.9$, $\rho_2 = 0.999$, $\delta = 10^{-8}$ work properly, although in some cases it is necessary to further tune the learning rate λ. The learning rate λ provides here a maximum for all individual learning rates for the parameters as these are adapted throughout the process. A decaying learning rate is therefore often not necessary for Adam, although it can be used to force the individual learning rates to decrease.

There does not seem to be a best optimization technique for machine learning. In research by Schaul et al. [83] many optimization techniques were compared for different learning tasks. Although they did find that adaptive learning rate algorithms seem to be more robust than other algorithms, a winner did not arise. Hence choosing the optimization algorithm is left to the designer and different algorithms can be compared to find which works best for the task. Many current machine learning packages include the named optimization algorithms.

3.3.5. Regularization

It is essential to prevent overfitting as much as possible in order to achieve good generalization properties of the model. One of the main tools to prevent this is regularization, in Section 3.3 we have already seen the L_1- and L_2-regularization on the parameters of the model. With regularization on the model parameters, the L_1 or L_2-norm of the learnable weights is added to the loss function. The total costs with loss function and regularization are shown in Equation (3.10).

$$J(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{p}) = L(\hat{\mathbf{y}}, \mathbf{y}) + \gamma \Gamma(\mathbf{p}). \tag{3.10}$$

The optimization algorithm then also requires the norm of the weights to be small, which can prevent the sizes of the weights becoming too large.

Two other common methods that achieve regularization, without changing the loss function, are early stopping and dropout. Early stopping is an intuitive approach based on the situation in Figure 3.2. When using early stopping, one keeps track of the validation loss and stops the training when the validation loss starts to increase again. In practice, the validation loss will show oscillations due to the use of batches and gradient approximation. Hence the training is stopped if the validation loss has not decreased within a fixed number of passes. Often the model parameters that have given the best validation set performance are saved before continuing training. It is advised to incorporate at least early stopping as a regularization technique in the training algorithm.

Algorithm 4: The Adam algorithm

Result: Updated parameters $\mathbf{p}$.
Input: Learning rate coefficient λ, initial parameters $\mathbf{p}$, $\delta > 0$ (required for numerical stabilization), exponential decay rates for moment estimates ρ_1 and ρ_2 in $[0, 1)$.
Initialize 1st and 2nd moment variables $\mathbf{s} = 0, \mathbf{r} = 0$ and timestep $t = 0$.
while *stopping criterion not met* **do**
  Sample a minibatch of m samples from training set $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_m, \mathbf{y}_m)\}$.
  Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\mathbf{p}} \sum_i J(\hat{\mathbf{y}}, \mathbf{y})$
  $t \leftarrow t + 1$
  Update biased first moment estimate: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$
  Update biased second moment estimate: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2)\mathbf{g} \odot \mathbf{g}$
  Correct bias in first moment: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1-\rho_1^t}$
  Correct bias in second moment: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1-\rho_2^t}$
  Compute update: $\Delta\mathbf{p} = -\lambda \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$ (operations applied element-wise)
  Apply update: $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.
end

The other frequently used regularization technique is dropout. During each training pass, certain nodes are dropped out with probability p, i.e. their outputs are replaced by 0, meaning that they are not used for prediction or training for that pass. Each batch different nodes will be dropped out, changing the network architecture slightly. This technique prevents co-adaptation where some weights will have large predictive capabilities and others have low predictive capabilities. The application of dropout makes it possible to use the full potential of large and deep neural networks. Common values of p are 0.1, 0.2 or 0.5. During evaluation on the validation or test set, dropout probability is always 0, ensuring that all nodes are used when evaluating.

3.4. Validation & Testing

In this section, the importance of validation and testing and the difference between these two are discussed. When the network has been trained, it is important to know how well it performs on samples it has not seen before to get an idea of how accurate its future predictions will be. To this end, the data is split in advance into a training and test set. The division ratio can vary depending on the amount of data, commonly used splits are 90%-10%, 80%-20%, or 70%-30% of the total set. The model is trained solely on the samples in the training set and can afterward be tested on the test set. The performance measure assists in deciding whether the network can perform the task accurately enough. Testing on unseen samples shows whether the network has learned the relations in the data or whether it has simply memorized the outputs.

Choosing a performance measure that fits the goal you want to achieve is important. Sometimes the loss function can be used as a performance measure, although the loss function and the performance measure should not be confused. As explained the loss function gives a measure for the difference in predicted and true output and is used for the optimization of the learnable parameters. However, often the objective is to optimize a specific performance measure on the test set. This measure can be completely different and can be unsuitable for optimization purposes. For illustration purposes consider a neural network with the goal to detect a rare disease. The loss function can be minimized when predicting every case to be 'healthy'. For a good performance, we need the model to actually capture the rare disease and not just minimize the loss function. Even when the same functional is used for both the loss function and the performance measure there might arise differences as the performance measure is computed on a separate test set.

The hyperparameters are tuned in order to improve the performance of the trained model on the test samples. However, by using the performance on the test set to tune the parameters these samples are indirectly seen by the network. The test set is no longer independent from the trained model as part of its information is contained in the tuned hyperparameters and hence in the trained model, which is called data or information leakage. The performance measure on the test set does not give the true

generalization performance of the model and it can be that the model performs worse in real life than the performance on the test set tells us. To prevent this and to still be able to tune the hyperparameters the training set is further split into a training-training set and a training-validation set (80%-20% or 90%-10%) or in short training and validation set.

Note that datasets are often organized or ordered and therefore it is important to shuffle the data before splitting into the training and test sets to prevent unwanted biases arising during learning.

3.4.1. Cross-validation

For small datasets, it is unwanted or not feasible to keep the test set completely separated until the end and use a validation set for tuning the hyperparameters. In that case, cross-validation can be used. This technique allows using all data samples while still acquiring an approximation of the generalized performance of the trained model [31].

In the common k-fold cross-validation, the dataset is split into k disjoint sets of almost equal size. These disjoint sets should be constructed randomly to prevent unwanted bias. The training of the network is performed k times, each time training on $k-1$ subsets and validating on the kth subset such that each subset is once the validation set and $k-1$ times in the training set. This process results in k performance measures, the average of which gives the cross-validated performance. Common used values for k are $k = 5$ or $k = 10$ [14, 29]. In cases where a test set can be kept aside for the final performance, cross-validation can be useful to achieve more generalized performance results for tuning the parameters. In Figure 3.7 an example of 10-fold cross-validation is given, where a test set is held aside.

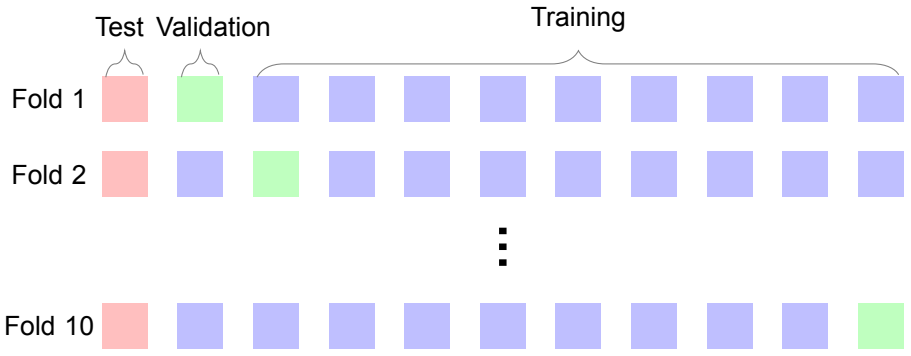


Figure 3.7: Example of k-fold cross-validation with $k = 10$ with separate test set. The dataset is split in a separate testing set and a training set divided in 10 random, equally sized subsets. The neural network is trained on $k-1$ subsets and validation on the remaining subset. The final performance is the average over all folds.

As a last remark, it can be helpful for the designer to consider if there are special cases that the network might come across during application, that aren't present in the (current) training set. Testing specifically on these special cases can give valuable information about the generalizing properties and performance of the model.

3.5. Data generation

The dataset is an important aspect of machine learning as it is the core ingredient of the learning process. The network only learns the relations that the data describes, which means that if the data is biased or unbalanced it is likely that the network will be biased or unbalanced. If the dataset is not large enough, it might not be possible for the network to learn all the relations in the data. In this section, important criteria for a machine learning dataset are discussed. In Section 3.6 data processing techniques such as scaling and principal component analysis are explained.

The dataset for training the neural network is generated by a numerical model. Using a numerical model to generate the data gives us control over the dataset. To the end of generating a proper dataset for machine learning purposes, the following criteria are taken into account:

- Relevancy: Naturally, the data should be relevant and the inputs must have a relevant correlation with the targets. The task of our neural network is to predict the output of the numerical model, hence as inputs and targets for the neural network we use the inputs and outputs of the numerical model respectively.

- Variability: The dataset must have enough variability in the input and target values in order for the network to find the correct relations and to prevent overfitting. Important criteria are that the parameter values are realistic and give stable solutions.

- Formatting: The data should be in a format that can be handled by the neural network. Often in machine learning, data is handled as tensors. An advantage is that the tensor datatype can easily be transferred to the GPU for fast computations.

- Accessibility: During training, the data needs to be accessed every epoch, and hence it should be easily accessible. In order to achieve this, a custom Dataset class is developed that contains the proper transforms and the input and target tensors. If input and target tensors are formatted with one of the axes for sample definition, the samples can easily be accessed by indexing.

- Size: The number of samples is important as the network needs to experience enough examples to learn proper generalizations. The minimum size needed for a good performance is highly dependent on the complexity of the task. In advance, it is unsure how many samples are necessary to achieve a good performance. Data generated by a numerical model has the advantage that more samples can be generated relatively easily, with available time and computational resources as the only restrictions. However, it must be noted that using more samples than necessary is not preferred, especially for the two- and three-dimensional model as computational time per sample increases rapidly.

- Scaling: For efficient learning, it is necessary to scale the values to a fixed range, e.g. between 0 and 1 or -1 and 1. One can imagine that it is hard to find proper weights during training when some input values are of order 10^{-3} and others of 10^{3}. The scaling of the values is considered after the generation of the set. Furthermore, the input values also affect the computed step size in gradient-based optimization algorithms.

3.6. Data processing

3.6.1. Scaling

Scaling of the input and output values is necessary for the learning process to ensure all values are of the same order. Large variations in the order of the values can make it hard for the network to learn the right weights. Well known scalers are MinMax scaling, scaling between 0 and 1, and standardization, scaling such that mean = 0 and std = 1. Both scalers are shown in Equation (3.11).

$$x_{\text{minmax}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad x_{\text{standard}} = \frac{x - \mu_x}{\sigma_x}. \tag{3.11}$$

Standardization actually computes the Z-score with respect to the sample mean and sample standard deviation. It is important to note that each input and output feature are separately scaled over all samples. Furthermore, the scalers should be fitted on the training set only, otherwise the scaling would already contain information on the validation samples. Lastly, one should not forget to inverse transform the predictions made by the network to obtain the original predicted values.

3.6.2. Principal Component Analysis

Principal Components Analysis (PCA) has been proven to be an effective method to reduce the dimensionality of a problem and increase interpretability without much loss of information [45]. In this section, the mathematical basics of PCA are explained and its application in machine learning is discussed.

Mathematical background

PCA computes an orthonormal basis for the data, which is used to project the data into a lower dimension. The orthonormal basis is defined by the principal component vectors and the largest principal

components provide the dimensions on which to project. First, the data is normalized to prevent unwanted influence from the order of each dimension, see Equation (3.12). Here μ_X is the column-wise mean of X.

$$\tilde{X} = X - \mu_X. \tag{3.12}$$

PCA then uses the Singular Value Decomposition (SVD), see Equation (3.13) to determine the principal components.

$$X = U\Sigma V^T \tag{3.13}$$

The SVD is a matrix decomposition method that contains the singular values σ_i in the diagonal matrix Σ, the left singular vectors $\mathbf{u}_i$ in matrix U and the right singular vectors $\mathbf{v}_i$ in matrix V. We note that it is always possible to determine an SVD with the singular values in descending order in matrix Σ. Since X is a real matrix, the matrices U and V are real and each provides an orthonormal basis for the matrix X. The singular values describe the explained variance by each principal component and hence can be used to determine how many components or dimensions are necessary to capture the necessary information in the data. The orthonormal basis along which to project is determined by the singular vectors corresponding to the chosen singular values.

Utilizing the explained variance ratio of the singular values, it is easy to determine how many components are needed in order to contain a specific amount of information. The explained variance ratio of each principal component is computed by:

$$EV_{\sigma_i} = \frac{\sigma_i}{\sum_{i=1}^{N} \sigma_i} \tag{3.14}$$

The number of components can be then be determined by adding principal components until the total explained variance is above a certain threshold. The data matrix X is then projected on the lower-dimensional space, described by an orthonormal basis corresponding to the chosen principal components.

Application in machine learning

The application of PCA in machine learning can contribute to a lower computational expense and a higher performance of the model. The reduction of dimensionality can make it easier for the network to learn the representation. Furthermore, a lower dimensionality increases the speed of the network during training and can prevent memory overload for large datasets. Often these advantages can be obtained without much loss of information. However, there are also some limitations when using PCA in machine learning algorithms [45]:

1. Model performance: Although the application of PCA can lead to an increase in performance and efficiency of a neural net, it can also lead to a reduction of performance for datasets that have low feature correlation or do not satisfy the linearity assumption. Care should be taken that enough information is preserved in the lower dimension.

2. Classification accuracy: Classification might be harder to learn if the information that distincts two classes is contained in the small principal components.

3. Outliers: PCA is sensitive to outliers, which can decrease learning performance.

4. Interpretability: The interpretability of the loss function decreases. For example the magnitude of the MSE of the (weighted) principal components gives us less information than the MSE between prediction and target directly.

In terms of training and validation, it is important to fit the PCA transform on the training set to prevent information-leakage from the validation set. Fitting on both training and validation data could allow information from the validation set in the fitted transform which would decrease reliability and representability of the validation set with respect to the test set.

By definition, transforming through principal components provides information on which parts of the data are most important. This knowledge can be used to enhance the learning of the neural network by adapting the loss function [20]. The loss function can be adapted to a weighted loss function by multiplying each component with its explained variance ratio, described in Equation (3.14).

The weighted MSE loss function, described in [20], is shown in Equation (3.15), where N_c is the number of chosen components, M is the total number of components, $y_i[j]$ and $\hat{y}_i[j]$ are the respective true and predicted value of the j^{th} feature of the i^{th} sample.

$$\sigma\text{-weighted MSE} = \sum_{i=1}^{N_{\text{samples}}} \sum_{j=1}^{N_c} \frac{\sigma_i}{\sum_{k=1}^{M} \sigma_i} (y_i[j] - \hat{y}_i[j])^2. \tag{3.15}$$

Utilizing this knowledge ensures that the network puts more focus on learning the largest principal components.

3.7. Performance measures

In this section, the performance measures used to determine whether one network is better than the other, are described. The performance measures include the R^2-score, the average relative error, the average L_2-norm and the average Relative Root Mean Squared Error (aRRMSE).

One of the reported measures is the coefficient of determination R^2, also called R^2-score. This score gives the proportion of variance in the output variables that is predictable from the independent input variables and hence is a measure for the goodness of fit. The R^2-score commonly returns a value between zero and one, where one indicates perfect prediction and zero indicates that the model always returns the expected value. The score can return a negative value, which shows the prediction of the model is worse than always returning the expected value. When dealing with multiple output values the uniform average of their scores is given. The R^2-score is given in Equation (3.16), where y_i denoted the true output, $\hat{y}_i$ the predicted output and $\bar{y}$ denotes the average true output for $i = 1, \ldots, N$ samples.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}. \tag{3.16}$$

Frequently used performance measures for (multi-target) regression problems include the average relative error, the Mean Squared Error (MSE) and the average Relative Root Mean Squared Error (aRRMSE) [4]. Although the average relative error is an easily interpretable performance measure it is not suitable for all targets. Some target values cross zero with a high gradient, resulting in high relative errors, while the prediction can prove to be accurate. The average relative error is given in Equation (3.17), where d is the total number of targets and N is the total number of samples.

$$a\delta = \frac{1}{N} \sum_{i=1}^{N} \delta = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{d} \sum_{t=1}^{d} \frac{\left| y_t^{(i)} - \hat{y}_t^{(i)} \right|}{y_t^{(i)}}. \tag{3.17}$$

The aRRMSE is a relative performance measure that can be used in cases where the relative error is not suitable and is given by Equation (3.18). According to Despotovic et al. [21], the performance of the model is excellent when the RRMSE is smaller than 0.1, good when the RRMSE is between 0.1 and 0.2, fair if it is between 0.2 and 0.3 and poor if it is above 0.3.

$$aRRMSE = \frac{1}{d} \sum_{t=1}^{d} RRMSE = \frac{1}{d} \sum_{t=1}^{d} \sqrt{\frac{\sum_{i=1}^{N} \left(y_t^{(i)} - \hat{y}_t^{(i)} \right)^2}{\sum_{i=1}^{N} \left(y_t^{(i)} - \bar{y}_t \right)^2}}. \tag{3.18}$$

The average L_2-norm gives an estimate of the average total distance between the predictions and the targets for each sample and is given in Equation (3.19). It should be kept in mind that the aL_2-norm is dependent on the magnitude of the simulations and the number of target values for each sample.

$$aL_2 = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}||_2 = \frac{1}{N} \sum_{i=1}^{N} \sqrt{\sum_{t=1}^{d} |y_t^{(i)} - \hat{y}_t^{(i)}|^2}. \tag{3.19}$$

4

Surrogate Model

In this chapter, neural network surrogate models are trained for prediction of relevant outputs of the morphoelastic models. It is assumed that the simulations from the numerical model are the true values and inaccuracies in these solutions are not taken into account. In Sections 4.1 - 4.3, surrogate models are developed for the one-dimensional model and in Section 4.4, a preliminary study is performed for the two-dimensional model. The neural networks are trained for the prediction of the relative surface area of the wound and the strain energy over time. Different approaches are considered and tested on the test set. Firstly, the chosen parameters values and the constructed dataset are discussed. Subsequently, the results for the prediction of the RSAW and the strain energy are discussed. For the two-dimensional model, a surrogate is trained for the movement of the wound edge over time to find localized contractions. Lastly, Section 7.1.1 gives the conclusion on the performance of neural network surrogates for the morphoelastic models.

4.1. Dataset - 1D morphoelastic model

In order to train the neural network, a dataset needs to be generated from the simulations performed by the one-dimensional morphoelastic model. The inputs for the neural network are equal to the input parameters that vary between simulations. The targets or outputs for the neural network are the outputs of the morphoelastic model. The model has 25 input parameters that can vary over the domain or between patients/simulations. All other constant input values are ignored as the neural network is able to learn these implicitly. To achieve a well-varied dataset, a range of acceptable values is defined for each of the input parameters. For each simulation, the parameters are drawn uniformly from these ranges. The advantage of drawing from a uniform distribution is that a larger variance in the dataset is achieved as all combinations of parameter values can be considered. The disadvantage is that drawing from a uniform distribution can result in unrealistic combinations since certain parameters can be correlated in real life. For some combinations of parameters, the finite element simulations were no longer stable and when this occurred the simulation was discarded and the parameter combination was saved separately. When generating future datasets, it should be ensured that the parameters fit the stability conditions derived in [23]. Table 4.1 shows the ranges for the parameters that varied between simulations. For means of reproducibility, the fixed parameters are displayed in Table 4.2. In Appendix A the meaning of the parameters and the choices for the parameters ranges are explained. The choices for the parameters are based on a literature study performed by Egberts et al. [22].

Each simulation computes the results on a domain of 10 cm, i.e. $\Omega = [-10, 0]$, with a uniform spatial grid of 202 grid points. We simulate 365 days with a time step of one day. The initial wound has length L , i.e. $\Omega^w = [-L, 0]$), with initial conditions as described in Section 2.1. As discussed in Chapter 2 the numerical model computes four constituents (N, M, ρ, c), three mechanical values (u, v, ϵ), the relative surface area (RSAW), and the strain energy (E_{strain}). We consider the relative surface area and the strain energy to be the important outputs of the model as these values provide direct information on skin contraction.

In total 6120 simulations are computed, of which nine were unstable. Therefore, the final dataset contains 6111 samples. Table 4.3 shows the data that is saved in the dataset with the respective

formats. Here the attribute 'Inputs' contains the 25 parameter values shown in Table 4.1. In Section 3.4 the importance of a validation and test set is discussed. For the one-dimensional dataset, we chose to use the common 80%-20% train-test split to ensure a large enough test set. Hence the training set contains 4889 samples and the test set contains 1222 samples. For the validation, 10-fold cross-validation is performed using a 90%-10% split of the training data.

Parameter	Range
D_F	$7.6167 \cdot 10^{-7}$ - $1.86624 \cdot 10^{-6}$
χ_F	$(2-3) \times 10^{-3}$
D_c	$(2.22-3.2) \times 10^{-3}$
r_F	$0.832 - 0.924$
$r_F^{\max}$	$2-3$
k_F	$5.4 \times 10^6 - 1.08 \times 10^7$
$\bar{N}$	$(1-2.5) \times 10^4$
$\bar{\rho}$	$0.0975 - 0.1200$
δ_N	$0.0119 - 0.0231$
δ_M	$0.06 - 0.0885$
δ_c	$(0.354-0.693) \times 10^{-3}$
δ_ρ	$(4-6.075) \times 10^{-6}$
κ_F	$10^{-7} - 10^{-6}$
$\tilde{c}$	$(1-5) \times 10^{-8}$
a_c^I	$(0.9-1.1) \times 10^{-8}$
a_c^{II}	$(0.98-1.02) \times 10^{-8}$
a_c^{III}	$(2-2.5) \times 10^8$
a_c^{IV}	$(0.8-1.2) \times 10^{-9}$
k_c	$(0.5-0.6)\delta_c \bar{\rho} a_c^{II}$
ρ_t	$0.89 - 1.29$
μ	$10 - 1000$
E	$350\text{-}\frac{300}{\sqrt{\bar{\rho}}}$
ξ	$(4.38-4.42) \times 10^{-2}$
ζ	$(1-9) \times 10^2$
L	$3-5$

Table 4.1: Ranges for the parameters that vary per patient in the one-dimensional dataset. Explanation of the parameters and ranges can be found in Appendix A.

Parameter	Value
$k_\rho^{\max}$	10
$\bar{M}$	0
$\bar{c}$	0
q	$\frac{[\log(\delta_N) - \log(r_F(1-\kappa_F \bar{N})]}{\log(\bar{N})}$
η^I	2
η^{II}	0.5
$\tilde{N}$	$0.2\bar{N}$
$\tilde{\rho}$	0
k_ρ	$\delta_\rho \bar{\rho}^2$
R	0.995

Table 4.2: Parameter values that are fixed for all patients in the one-dimensional simulations. Explanation of the parameters and values can be found in Appendix A.

Key	**Size**
Inputs	$N_{\text{samples}} \times 25$
RSAW	$N_{\text{samples}} \times 366$
StrainEnergy	$N_{\text{samples}} \times 366$
N	$N_{\text{samples}} \times 366 \times 202$
M	$N_{\text{samples}} \times 366 \times 202$
c	$N_{\text{samples}} \times 366 \times 202$
ρ	$N_{\text{samples}} \times 366 \times 202$
u	$N_{\text{samples}} \times 366 \times 202$
v	$N_{\text{samples}} \times 366 \times 202$
ϵ	$N_{\text{samples}} \times 366 \times 202$
x	$N_{\text{samples}} \times 366 \times 202$

Table 4.3: One-dimensional dataset - keys and sizes.

4.2. RSAW prediction- 1D morphoelastic model

The Relative Surface Area Wound (RSAW) is one of the main outputs from the model, and therefore we start by training a network to predict this output from the 25 input parameters. It is important to keep in mind that minimum RSAW, i.e. maximum contraction, and the asymptotic/final value of RSAW, i.e. final contraction, are important values of the RSAW distribution. All simulations are run with 10-fold cross-validation. Using cross-validation ensures that all samples, including outliers, are considered in one of the validation sets. The cross-validation allows us to compute mean and standard deviations of the performance measures which can indicate how well the model can generalize. In case the standard deviation over the performance of all validation sets is small, this shows that the network can predict all samples equally well. A large deviation indicates that the network has difficulty predicting certain samples leading to a lower performance when these samples are in the validation set.

The choice of hyperparameters is important for the performance of the model, and multiple combinations have been tested to find a suitable combination. As the goal of this research is to investigate the possibilities of a neural network surrogate the hyperparameters are not fully optimized, but merely tuned to a good performance. To the end of tuning the hyperparameters, different loss functions, optimizers, architectures, and learning rates are used and compared on their performance. More information on the experimental training runs for the hyperparameter tuning can be found in Appendix C.

In this section, the results are shown from training the network to predict the RSAW over 365 days based on a suitable choice of hyperparameters. Furthermore, the use of Principal Components Analysis (PCA) is investigated to reduce the dimension of the output. In Section 4.2.3 the effect of the size of the training set is investigated to find if the current dataset contains enough samples. A final network is trained on both training and validation data and tested on the test set and nine unstable samples. A second approach to predict RSAW is studied by training a neural network to predict the displacement.

4.2.1. Neural network

First, we train a neural network to predict the RSAW for all 365 days. The neural network that gives a good performance consists of a two-layer MLP with 100 nodes in each layer and ReLU activation units. The network is trained with MSE loss function and the Adam optimizer, using a learning rate 0.01 with decay factor 0.99 every epoch. The network is trained for max 150 epochs, using early stopping as a default regularization technique. This regularization technique stops the training when the validation loss has not decreased to prevent overfitting. This network is trained on the standardized one-dimensional-dataset processing the data in batches of 64 samples. The training of the network is performed on the Graphics Processing Unit (GPU) to decreases the run time. An overview of the chosen hyperparameters can be found in Table 4.4.

Type NN	MLP
No. neurons	100/100
Activation function	ReLU
Loss function	MSE
Optimization	Adam
Learning rate	0.01
Learning rate decay	0.99
Max #epochs	150
Regularization	Early stop (20)
Data processing	Standardization

Table 4.4: Neural network hyperparameters for RSAW prediction.

The overall results for this network are shown in Figures 4.1, 4.2, and in Table 4.5. Figure 4.1 shows the best and the worst prediction in terms of the MSE, the relative error at each point for the worst prediction, and the relation between the predicted and target values for all samples in one of the 10 cross-validation sets. In case of perfect predictions, this scatter plot would show exactly the $y = x$ - graph, which is shown for comparison. In Figure 4.1 it can be seen that in the best case scenario the prediction is almost indistinguishable from the target. The worst prediction in the validation set is higher than the target value after 50 days, but it predicts the minimum value at the correct day. We find that the relative error increases to 15% and converges to about 6% for the final contraction.

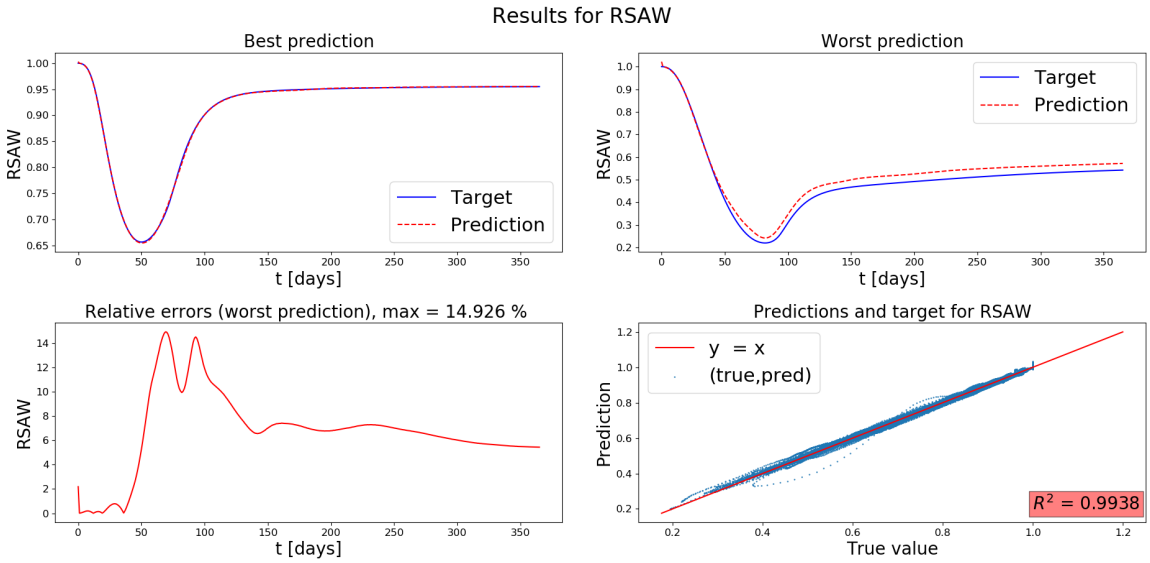


Figure 4.1: Results Neural Network (NN) for RSAW prediction. Upper two graphs show predictions and targets for the best and the worst prediction in terms of MSE. Left bottom graph shows the relative error for the worst prediction and right bottom graph shows the relation between the predictions and the targets, the line 'y = x' for comparison and the R^2-score.

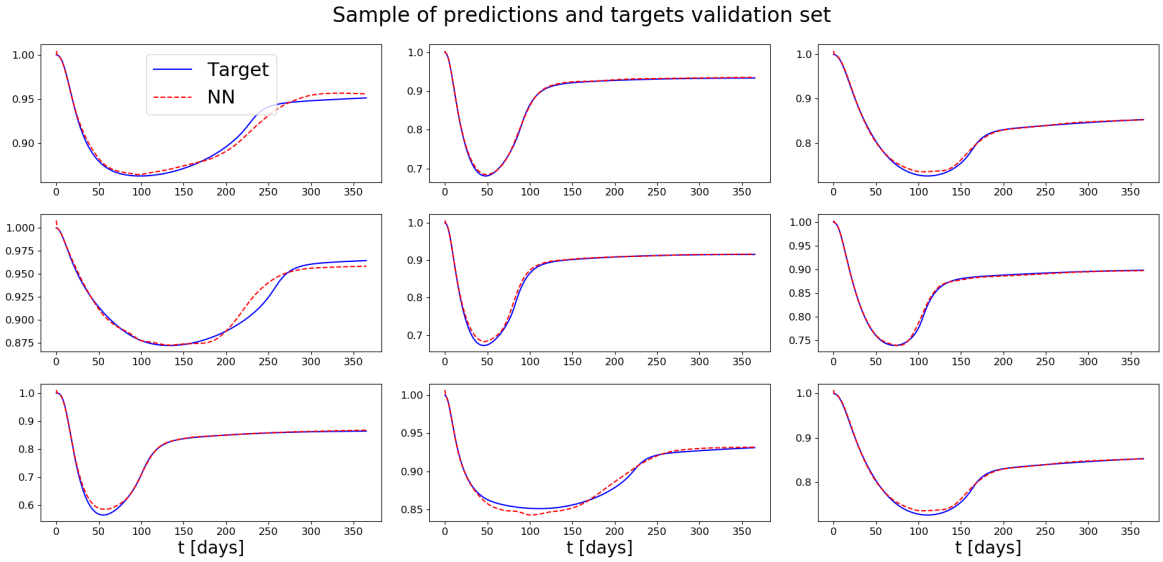


Figure 4.2: Random chosen RSAW samples from the validation set and the corresponding predictions from the neural network.

The distribution of the predictions and targets is close to the $y = x$ line, which shows that in general the predicted values are close to the true values and therefore the predictions are accurate. Figure 4.2 shows nine random samples from the validation set and the network predictions. For these nine samples, the neural network prediction is close to the target value. To substantiate our observations, Table 4.5 shows the performance measures as well as training and validation time. The R^2-score is a measure of the goodness of fit and gives a measure of how well the variation in the output is described by the variation in the input. A R^2-score often returns a value between 0 and 1, where 1 indicates perfect predictions. However, the R^2-score can give negative results when the predictions are worse than always returning the expected value. The cross-validation trials return a mean R^2-score of 0.9941 with a standard deviation 0.0004. The result is close to one and therefore shows that the predictions are accurate. The standard deviation of 0.0004 shows that the 10 validation sets all return a similar R^2-score all close to one. The aRRMSE is 0.0525, which is smaller than 0.1 and therefore the results

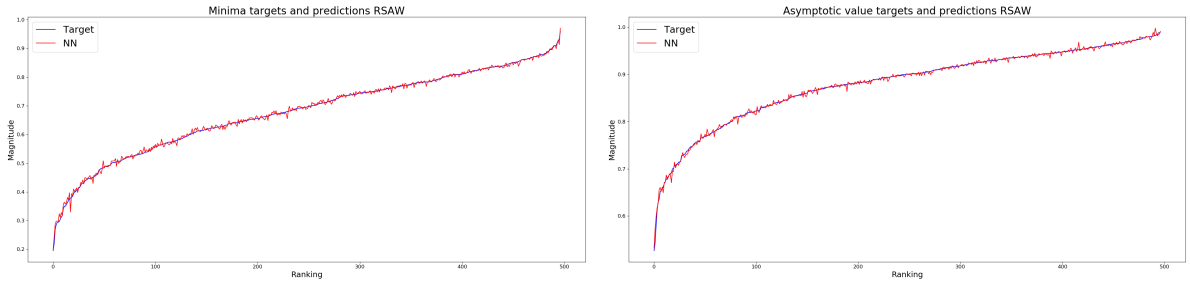
Performance measure	Value
R^2	0.9941 ± 0.0004
aL$_2$ norm	0.084 ± 0.004
aRRMSE	0.0525 ± 0.0024
aRelErr	$0.45\% \pm 0.03\%$
Training time	230 s
Validation time	0.0008 s

Table 4.5: General performance of neural network for RSAW (Table 4.4) on the standardized one-dimensional dataset. The mean and standard deviation of the performance measures over all 10 cross-validation sets.

are very good, as is explained in Section 3.7. The average relative error of the predictions of only 0.45% is low, which supports our claim that the neural network can accurately predict the RSAW.

Besides the overall performance, it is interesting to consider the performance of the network on the prediction of the minimum and the final value of the relative surface area as these are important characteristics of skin contraction. Considering the performance on these characteristics has the advantage that the values are easier to interpret than the overall performance. To that end, Table 4.6 contains the Mean Absolute Error (MAE) and the R^2-score for both minimum and final RSAW over the validation sets. To place the MAE in context also the general characteristics of the distributions are shown.

The Mean Absolute Error (MAE) of 0.0051 on the minimum RSAW is less than 0.6% of the range of values and less than 0.75% of the average value. The MAE shows that the predictions of the minimum are on average close to the true value and the fact that it is only 0.6% of the total range shows that the network can distinguish well where in this range the minimum is located for each sample. We visualize this concept in Figure 4.3(a) where the target minimum values are ranked from low to high and the predictions are shown for each target. The predictions in the figure follow the ranking of the target values closely showing that the network can accurately predict where in the range of values the minimum is located. For the final RSAW the MAE of the final RSAW is 0.43% of the range of values and 0.30% of the average value. Using the same arguments we conclude that the network can predict the final contraction well. The small standard deviation on the performance measures show that the generalization of the model is good, reaching similar performance on all validation sets.



(a) Minimum RSAW

(b) Final RSAW

Figure 4.3: Distribution of RSAW minima and final values ranked on the size of the true values.

	MAE	R^2	Min	Max	Range	Average
Min RSAW	0.0051 ± 0.00025	0.9976 ± 0.00028	0.090	0.958	0.868	0.689
Final RSAW	0.0026 ± 0.0001	0.9976 ± 0.00034	0.390	0.992	0.602	0.88

Table 4.6: Results for the minimum value of RSAW (max contraction) and the final value of RSAW (final contraction) averaged over the 10 cross-validation trials. Distribution of the values is provided for context of the Mean Absolute Error.

From the good overall performance measures, the small standard deviations, and the good performance on the minimum and final RSAW predictions, we conclude that the trained network can accurately predict the relative surface area of the wound. Lastly, it is noted that validation time is only 0.0008 seconds in which the network predicts approximately 480 samples. This is significantly faster than the numerical model, which takes 90 seconds for one simulation.

4.2.2. PCA

The network trained in the previous section needs to predict 365 values as output. Since many of these values are closely related, using only the larger principal components can reduce the dimension without reducing much of the information in the data. Predicting fewer values can improve the training and the performance as long as the important information is captured in the lower-dimensional components [20]. Furthermore, using lower dimensional targets and predictions reduces the required memory and the computational time.

To determine the number of principal components we consider the explained variance ratio, which shows how much of the variance of the data is described by the number of used principal components. An explained variance ratio of one means that all variance in the data is explained by the used components. In Figure 4.4 the explained variance ratio is shown against the number of principal components for the first 22 components of RSAW. The figure shows that the first component captures more than 80% of the total variance, the first five components capture over 95% of the variance and that the explained variance converges rapidly to one. The fast convergence to one indicates that the dimension can be reduced by only considering the largest components, though it should be checked what type of information is lost by ignoring the smaller components. As mentioned in Section 3.6.2 when discussing the limitations, important information for training a neural network can be found in the smaller components.

To that end, the effect of the transforms with a different number of components has been studied by transforming and inverse transforming the samples to find the loss of information. Figure 4.5 shows two representative samples that have been transformed and inverse transformed for PCA-transforms with an increased number of components. For the transforms with only two or five components, the RSAW distributions after transforming and inverse transforming have a different minimum and a different shape and therefore important information is lost. For the first sample, the transform with 14 components gives a small overshoot at the beginning and at the minimum, and the transform with 22 components results in a RSAW distribution that is almost indistinguishable from the original distribution. For the second sample, the transform with 14 components results in small oscillations around the true shape and the transform with 22 components gives a close to perfect resemblance of the original distribution. Based on these results, it is decided to use 22 components, or a 22-dimensional PCA-transform, before training.

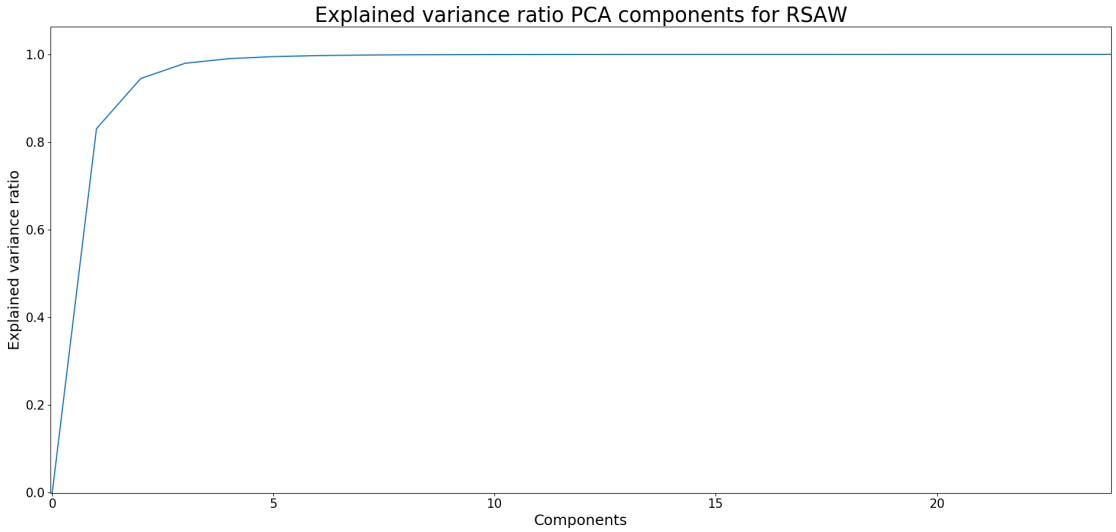
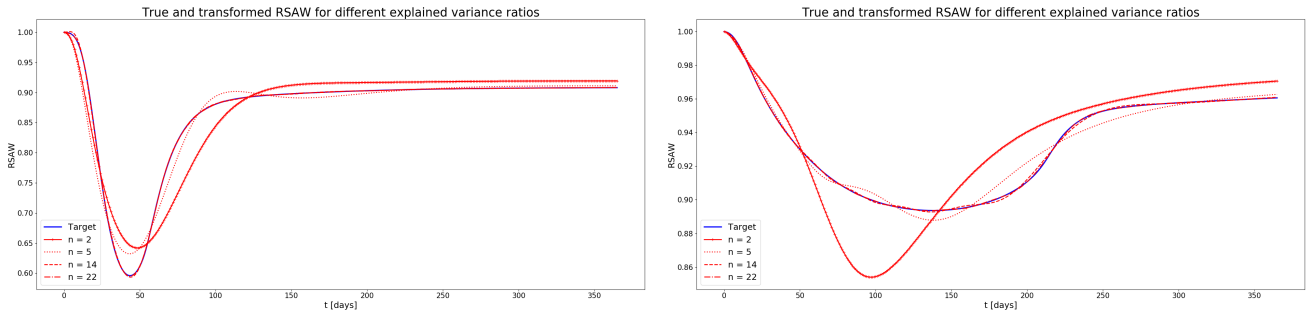


Figure 4.4: Explained variance ratio for increasing number of principal components for the RSAW distribution. Only the first 22 components of 365 are depicted.

To determine the effect of the PCA-transform on the performance of the neural network, a new neural network is trained to predict the transformed RSAW distributions. Here the same hyperparameters from the previous section are used, as shown in Table 4.4. The PCA-transform is fitted on the training set and the PCA target values are standardized over all the samples before training. The network is then trained to predict the PCA-components of the samples. The network is trained both with standard MSE



(a) Sample 1

(b) Sample 2

Figure 4.5: Inversely transformed - transformed RSAW distributions for fitted PCA-transforms with different number of components.

loss function and the dynamically weighted loss function based on the importance of each component as described in Section 3.6.2. The performance measures for training with both loss functions, can be found in Table 4.7.

Performance measure	MSE-loss	$\sigma-$weighted MSE
R^2	0.9544 ± 0.011	0.9930 ± 0.0005
aL_2 norm	0.3016 ± 0.0264	0.0804 ± 0.0035
aRRMSE	0.2049 ± 0.0261	0.0569 ± 0.0037
aRelErr	$1.8\% \pm 0.15\%$	$0.43\% \pm 0.02\%$
MAE minimum	0.0188 ± 0.002	0.0046 ± 0.00028
MAE asymptotic	0.0128 ± 0.001	0.0025 ± 0.0002
Training time	154 s	100 s
Validation time	0.0007 s	0.0007 s

Table 4.7: Performance measures for training a neural network with 22-dimensional PCA-transform on two different loss functions for hyperparameters defined in Table 4.4 over 10-fold cross validation.

In the table, it can be seen that the R^2-score for the $\sigma-$weighted loss function increases from 0.9544 to 0.9930, aRRMSE decreases from 0.231 to 0.057, and the average relative error decreases from 1.8% to 0.44%. For the predictions of the minimum and the final area, the MAE is approximately four times smaller if the $\sigma-$weighted MSE is used. In the table, it can also be seen that the standard deviations of the performance measures for the different cross-validations is lower for the model trained with the $\sigma-$weighted MSE loss, which indicates the model can generalize better. Since the average relative error, the aRRMSE and the aL_2 norm are 3-4 times higher for the MSE loss function and the R^2-score and standard deviations are significantly lower, it is concluded that it is better to use the $\sigma-$weighted MSE.

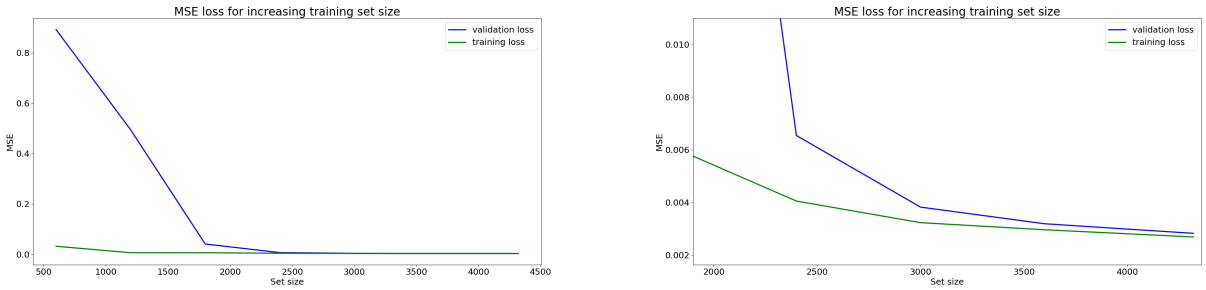
To determine the effect of applying the PCA-transform on the performance of the neural network, the performance is compared to the results of the neural network trained on the original data without the PCA-transforms in Table 4.5. The mean performance of both networks is similar and the standard deviations are of the same order, hence the use of the PCA-transform does not seem to lose valuable information. The PCA-transform does not improve the results, but this is not unreasonable as the original prediction was already very accurate. Lastly, we find that using the transform reduces the training time with a factor of 3 and reduces the required memory. Therefore, the PCA-transform is applied in the remainder of this chapter.

4.2.3. Training size

The size of the training set is important for the amount of variation the network can learn and hence how well it can generalize. In the ideal situation, the network would have a unlimited supply of samples such that all samples are independent and the model can stop training whenever the stopping criteria is met. However, achieving a dataset that approximates unlimited supply is unfeasible for almost every machine learning problem as it is expensive, time-consuming, or simply impossible to get such large

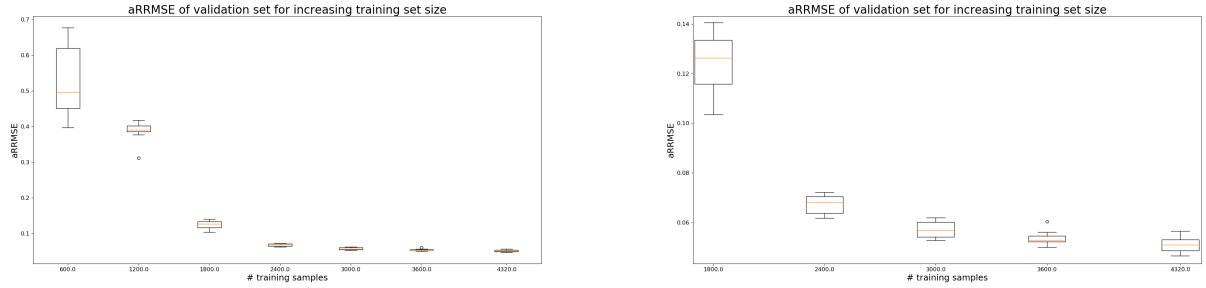
amounts of samples. It has been shown that the networks can be trained to achieve high accuracy by re-using the same samples multiple times, meaning that adding more samples might not lead to an improvement [31]. Since the generation of data from the numerical model is in general time consuming it is useful to investigate whether the current amount of data is sufficient or whether adding more samples can significantly improve the result.

To that end, the full training set is loaded and split using a 10-fold cross-validation. The network parameters from Table 4.4 and the 22-dimensional PCA-transform are used. The network is trained for each fold using an increasing number of samples, i.e. $[500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4320]$, and then evaluated on the respective cross-validation set. The results are combined for each size of the training set over all the folds. Figure 4.6 shows the learning curves, i.e. average training and validation loss, against the size of the training set. Figures 4.7 and 4.8 show box plots of the aRRMSE and the average relative error over the 10 folds for each size of the training set. The spread and average of the box plots give an indication of the generalization properties of the network for each size of the training set. A large spread shows that the performance of the validation set is dependent on the samples in that set, meaning the network does not generalize well to all types of samples.



(a) Full scale (b) Zoomed version

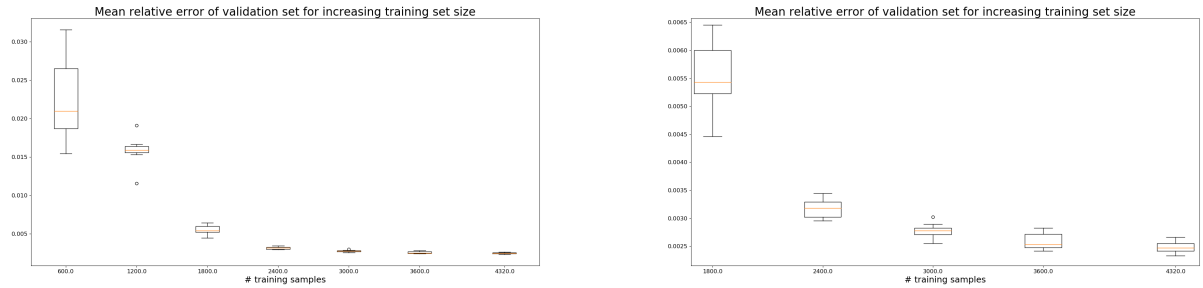
Figure 4.6: Averaged training and validation loss for increasing number of training samples.



(a) Full scale (b) Zoomed version

Figure 4.7: Boxplots of cross-validation aRRMSE values for increasing number of training samples.

In Section 3.1.2 it has been explained how to interpret the learning curves. For 600 training samples, the training MSE loss is approximately 0.025 network, whereas the validation loss is above 0.8. The large gap, with training loss far lower than the validation loss, shows that the network is overfitting the samples in the training set. It has only learned to interpret the training values and can not generalize to the unseen samples in the validation set. As expected, the gap between training and validation loss decreases by adding more samples to the training set. In the zoomed figure it can be seen that if 3000 samples or more are used, the gap is between the losses is smaller than 0.001, which shows that the network does not overfit and has learned to generalize to the samples in the validation set. The low MSE of approximately 0.003 indicates that the network is not underfitting. The decrease in training loss shows that not only generalization improved, but also the fit on the training set. The boxplots show the spread of the aRRMSE and the mean relative error for the 10 different cross-validation sets. Again using a few samples gives a large spread for the different validation sets, showing that the performance



(a) Full scale

(b) Zoomed version

Figure 4.8: Boxplots of cross-validation mean relative errors for increasing number of training samples.

network depends more on the samples in the validation set and therefore has lower generalization properties. After 3000 samples the boxplots have a larger overlap, from which we conclude that the performance measures do not improve significantly due to the added samples. Therefore, we can conclude that our training set is large enough to capture the current variation in the RSAW samples. It should be noted that the results from the training set size study are problem-specific and for a different problem the test needs to be repeated.

4.2.4. Final network & Test set evaluation

Different architectures and hyperparameters have been tested on the validation set, and we have found that the network with parameters from Table 4.4 with the 22-dimensional PCA-transform performs well on the validation set. The network is trained on both training and validation set for 150 epochs and it is tested on the test set to find the performance. First, the trained network is tested on the original constructed test set. Secondly, the network is used to predict the RSAW distribution for a small test set that contains input combinations that gave numerical instabilities. This last test is to verify how well the model is able to generalize to input combinations that gave stability issues in the morphoelastic model.

The test set contains 1226 samples. The trained network and fitted scalers are loaded and used to compute predictions on RSAW for all inputs in the test set. The general results are shown in Table 4.8 and Figure 4.9. The results for the minimum and final RSAW values are shown in Table 4.9 and in Figure 4.10a and 4.10b, respectively.

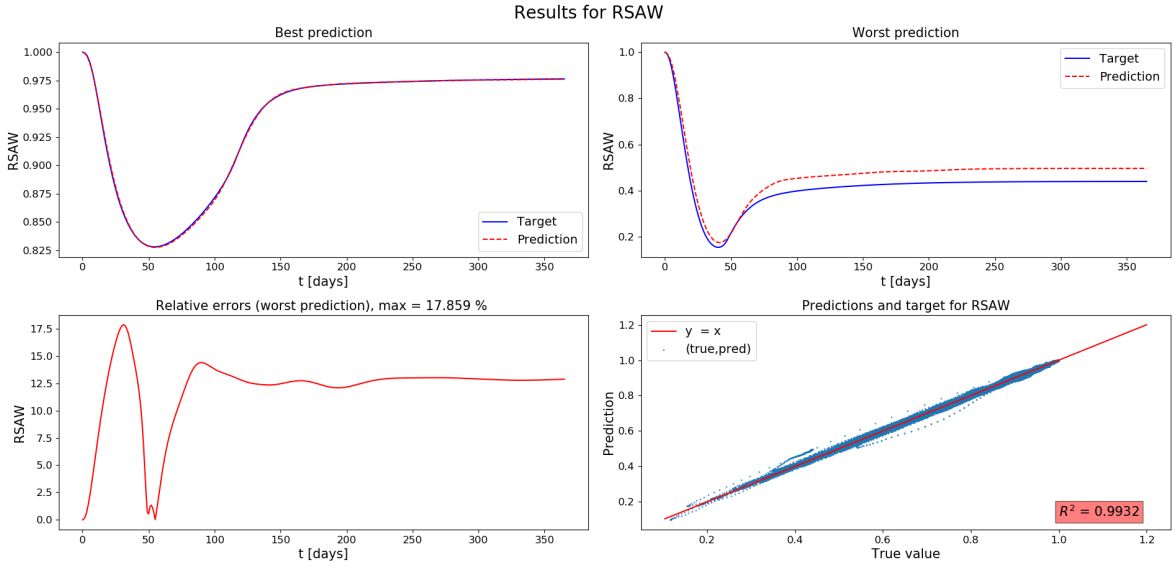


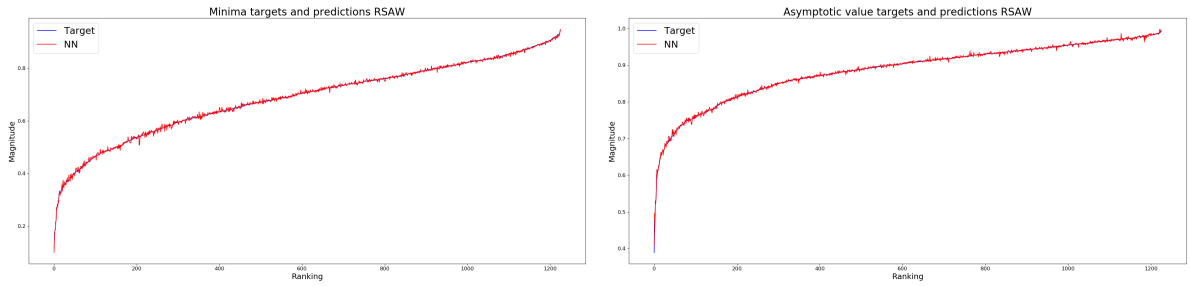
Figure 4.9: Results for the prediction of RSAW on the test set. Upper two graphs show predictions and targets for the best and the worst prediction in terms of MSE. The left bottom graph shows the relative error for the worst prediction and the right bottom graph shows the relation between the predictions and the targets, the line 'y = x' for comparison, and the R^2-score.

The performance on the test set is good with a high R^2-score of 0.9932, a low average relative error of 0.39% and average L_2-norm of 0.073. Comparing the results to the performance on the validation sets in Table 4.7, shows that the R^2-score and the aRRMSE are within one standard deviation of the respective mean values on the validation sets, which indicates that performance on validation and training set is similar. The average L_2 norm and the average relative error are slightly better than the validation set.

Performance measure	Value
R^2	0.9932
aL_2 norm	0.073
aRRMSE	0.055
aRelErr	0.39%

Table 4.8: General performance of the RSAW prediction on the test set

Considering the minimum and asymptotic value separately shows that the predictions are accurate with MAE less than 1% of the range, which is good, as is explained in the previous sections. The good performance indicates that the network has learned the characteristics and can generalize well to provide accurate predictions for unseen samples. The similar performance confirms the conclusion that the size of the training set was large enough to accurately learn the generalizations and that adding more data does not significantly improve the results, since the network has now been trained on both the training and the validation set.



(a) Minimum RSAW

(b) Final RSAW

Figure 4.10: Distribution of RSAW minima and final values for the test set, ranked on the size of the true values.

	MAE	R^2	Min	Max	Range	Average
Min RSAW	0.0038	0.999	0.116	0.944	0.829	0.686
Final RSAW	0.0023	0.998	0.389	0.944	0.555	0.886

Table 4.9: Results for the minimum value of RSAW (max contraction) and the final value of RSAW (final contraction) for the test set. Distribution of the values in the test set is provided for context of the Mean Absolute Error.

4.2.5. Exceptional test cases

During dataset generation, nine combinations of input values were encountered for which the morphoelastic model could not find a full solution due to numerical issues. The matrices during simulations become close to singular resulting in complex values and consequently NaN. It is likely that this is caused by the parameters not satisfying the stability conditions derived in [23]. For the first 100-150 simulated days, the numerical model gives results, though it is unsure whether these are accurate. It is interesting to see if the neural network can generalize well enough to provide predictions where the numerical model is not able to. Note that these samples can not be considered full test samples as there are no full targets available and it is therefore not possible to provide performance measures on the complete predictions. The predictions are compared to the stable beginning of the solution by the numerical model.

The neural network predictions for RSAW of these nine samples are shown in Figure 4.11, here the stable part of the results from the morphoelastic model are shown for validation. For samples 4, 6, 7,

and 8 the neural network prediction is close to the target and the prediction continues smoothly where the target stops. For samples 1, 2, 5, and 9 the neural network predicts the minimum at approximately the correct day, though the predicted minimum is approximately 0.03 to 0.05 lower than the expected minimum of the targets. From the observations, it seems that the model can give reasonable predictions even when the morphoelastic model encounters problems.

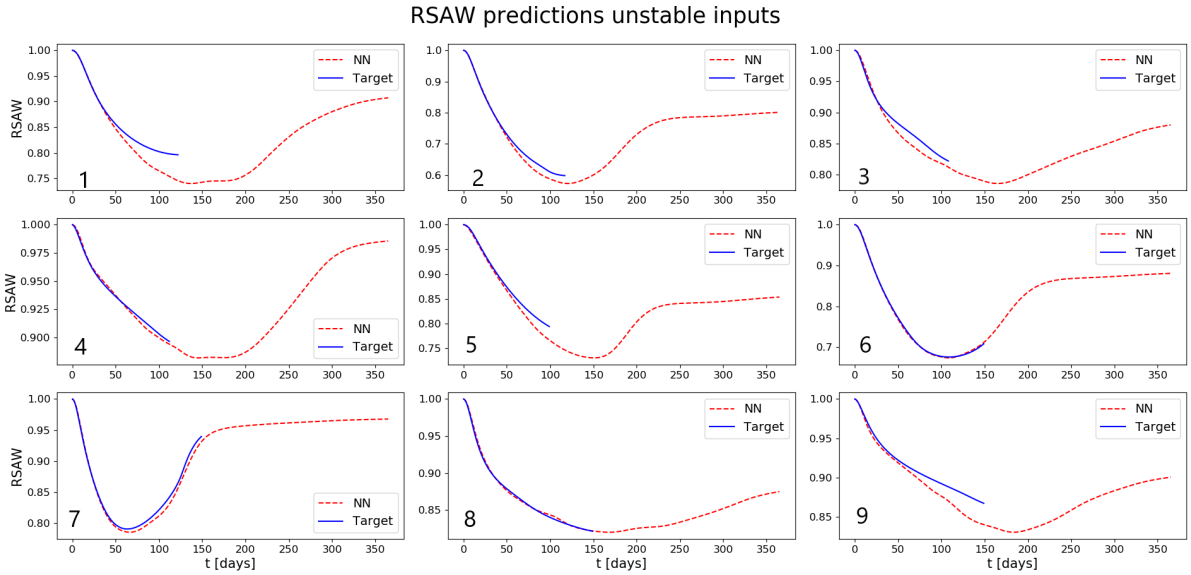


Figure 4.11: Predictions for inputs that gave incomplete results with the morphoelastic model.

4.2.6. RSAW prediction from displacement u

Besides training a neural network to predict the relative surface area directly, a different approach is studied using the displacement of the wound. In the morphoelastic model, the RSAW computation is a post-processing step using the initial size of the wound and the computed displacement u. Instead of learning the distribution of RSAW the network can be trained to predict the displacement u which can be used to perform the same post-processing step to obtain RSAW, which is shown in Equation (4.1), where L is the original size of the wound and x_b is the grid point that is on the edge of the wound.

$$\text{RSAW}_u(t) = \frac{|-L + u(t, x_b)|}{L}. \tag{4.1}$$

Using the displacement for the RSAW prediction instead of the direct prediction might have multiple advantages. When considering the future extension to two dimensions, the displacement can give more detailed information about the movement of the wound and therefore localized contractions. Furthermore, comparing the direct prediction with the prediction by the displacement can give an indication of the accuracy of the predictions. If the two predictions differ largely, the error in the prediction might be larger than when the two predictions are very similar. It is investigated whether the two differently trained neural networks could be formed into an ensemble of neural networks providing both an improved prediction and a measure of uncertainty.

The dimension of the displacement values is larger than the RSAW as it varies both over the place and in time, hence the dimension is 366×202 per sample. The larger dimension might make it more difficult for the neural network to give accurate predictions. Analogously to Section 4.2.2, a study on the explained variance and the effect on the sample is used to decide the number of components. From this study, it was chosen to use a 35-dimensional PCA-transform for the displacement values. We have chosen to train a three-layer MLP with 150 nodes in each layer for 150 epochs at most, using a learning rate of 0.008 with decay factor 0.99. The predicted displacement is used to determine the relative surface of the wound by considering the movement of the wound edge. The performance measures for the prediction of u are shown in Table 4.10. Figure 4.12 shows the displacement of the

wound on days 5, 25, 50, 150, and 365 for the best and worst prediction of u in the validation set. The figure shows that the prediction of the neural network follows the increase and decrease of the targets well over the days. In the worst prediction, the neural network predicts a lower maximum than the target, which is most clear on day 50. Even the differences for the worst prediction are still reasonable. The R^2-score of 0.9976 and the aRRMSE of 0.0426 support the observations that the predictions of the network are accurate. The average relative error of 5.67% is higher than expected based on the best and worst prediction and the other performance measures. The higher relative error is caused by the values of u close to zero the first few days and therefore does not give an accurate representation of the performance. Therefore, we can conclude that a neural network can also learn to predict the displacement well in time and over space.

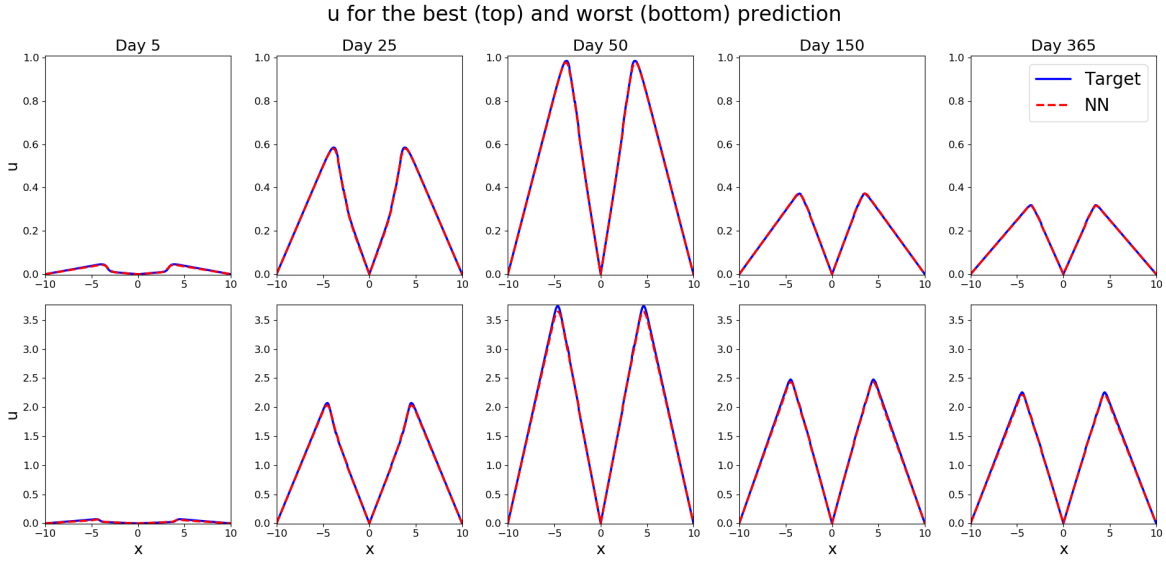


Figure 4.12: Worst and best prediction for u in terms of MSE at day 25,50,100,150, and 365.

The trained network is used to computed predictions for u on the test set and subsequently, the predictions of u are used to compute RSAW. The predictions of RSAW using the predicted displacement are named RSAW$_u$. The performance results for the prediction of RSAW$_u$ are given in the third column of Table 4.10. For comparison, the results for the test set predictions of RSAW by the neural network from Section 4.2.2, named RSAW$_{NN}$, are repeated in the table as well. The prediction of RSAW$_u$ is slightly worse as it has a higher relative error of 0.46% compared to 0.39%, a higher aRRMSE of 0.0639 compared to 0.0552, and a higher average L_2 norm of 0.083 compared to 0.073. The R^2-score on the other hand is slightly better. The differences in performance are small and it is unknown whether a better-trained network on u can improve the results of RSAW$_u$ to achieve the same accuracy as RSAW$_{NN}$. Based on these observations, no conclusions can be given on whether one method is significantly better than the other, though it can be concluded that both methods provide accurate predictions of skin contraction.

Performance measure	u	RSAW$_u$	RSAW$_{NN}$	Average RSAW
R^2	0.9976	0.9936	0.9931	0.9936
aL_2 norm	2.0	0.083	0.073	0.061
aRRMSE	0.0426	0.0639	0.0552	0.0492
aRelErr	5.67%	0.46%	0.39%	0.33%

Table 4.10: General performance for the prediction of displacement u, the prediction of RSAW using the prediction of u, the prediction of RSAW directly by the NN, and the averaged prediction on the test set.

Figure 4.13 shows nine random samples with the predictions of both methods and the R^2-score of the predictions, which indicates how well the two predictions fit each other. From the random samples, it is observed that both methods provide an accurate prediction. Furthermore, it can be observed that in

the right middle sample, the RSAW_{NN} prediction is better, and in the right bottom sample the RSAW_u prediction is better. In the middle top figure, one prediction method underestimates, while the other method overestimates the minimum RSAW. It is investigated whether combining the two methods in an ensemble by averaging their predictions improves the performance.

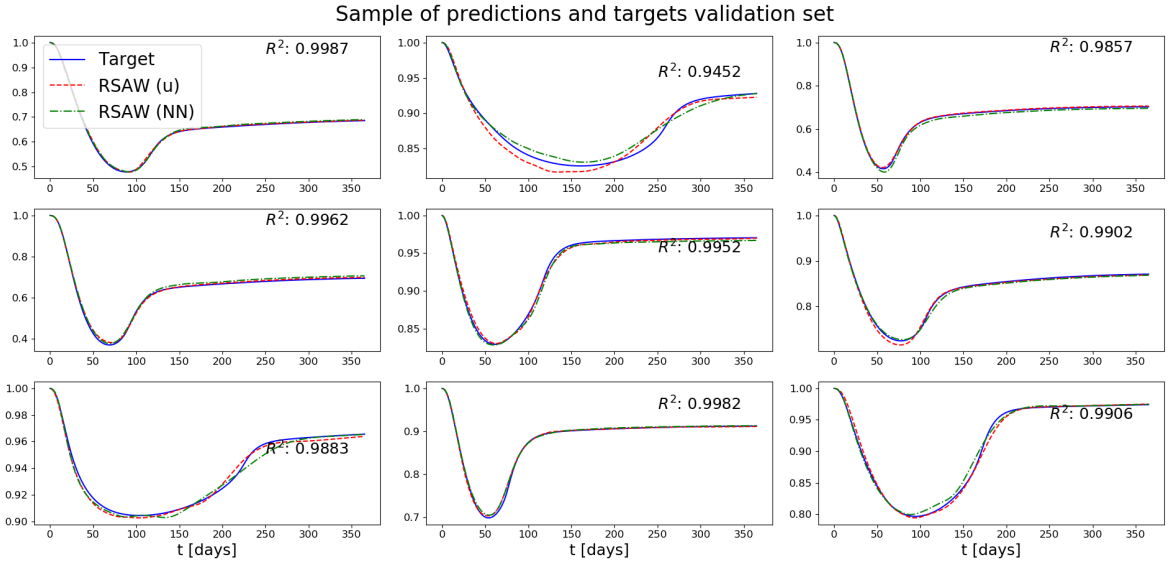


Figure 4.13: Nine random samples of the validation set with predictions RSAW_u and RSAW_{NN}. The R^2-score of the two predictions is shown for each sample.

The results for the averaged predictions are shown in the last column of Table 4.10. The performance measures for the averaged prediction are better than for the individual predictions, with a lower average relative error, aRRMSE, and aL_2 norm. The R^2-score is equal to the score of RSAW_u. The improved performance measures show that in this case a better prediction can be obtained by combining the two predictions. Figure 4.14 shows the relation between the mean absolute difference of the two predictions and the true MAE of the respective predictions. The Pearson correlation coefficient is reported, indicating to what extend the values show a linearity relation. The correlation coefficients of 0.4363, 0.585 and 0.3464 show that there is a moderate positive correlation between the mean absolute difference and true errors. This means that a higher mean absolute difference indicates, to some extent, a higher true error, though the correlation is not strong enough for good estimates of the uncertainty. It is concluded that using an ensemble of two differently trained neural networks can improve the predictions, though it cannot be concluded that this always leads to better performance. It might be possible to obtain the same results with the individual methods by using a different architecture or training.

4.3. Strain energy prediction - 1D morphoelastic model

Besides the relative surface area of the wound, the strain energy might give useful insight into the discomfort a patient experiences during the contraction process. High strain energy could indicate that the patient experiences a lot of pain or discomfort due to the forces on the skin. If there is remaining strain energy, it could indicate that the patient will experience long-term discomfort in the burned area. The strain energy is computed from the effective strain and the concentration of collagen as described in Section 2.1.3. First, a neural network is trained on strain energy data only and tested on the test set. Secondly, it is investigated whether one neural network can predict RSAW and strain energy simultaneously as these two distributions are related. Lastly, networks are trained to predict the effective strain and the collagen concentration, such that the strain energy can be computed from these predictions. The different prediction methods are combined in an ensemble to find if it can improve the prediction of the strain energy. It is studied whether the difference between the two predictions can give an indication of the uncertainty of the prediction.

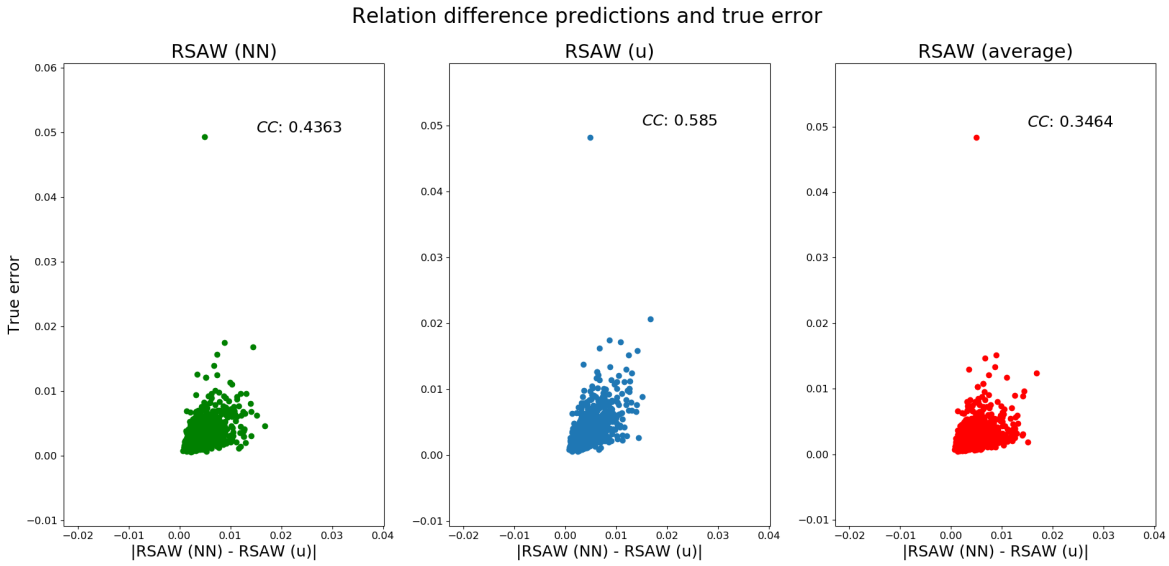


Figure 4.14: Correlations between mean absolute difference, |RSAW$_{NN}$ - RSAW$_u$| and the true MAE of the individual predictions.

4.3.1. Training

Analogously to the prediction of RSAW, the dimension of the strain energy vector is reduced using the largest principal components using the 25-dimensional PCA-transform. The choice for the number of components is based on the explained variance, the influence of the transform on the samples, and the influence on the training of the network. It was found that using 25 components hardly reduces the amount of information in the data. In order to find a suitable combination of hyperparameters, multiple combinations of hyperparameters have been tested, more information on hyperparameter testing can be found in Appendix C. Based on the hyperparameter experiments, it was chosen to train a two-layer network with 150 nodes in each layer and ReLU activation units. The main hyperparameters can be found in Table 4.11.

Type NN	MLP
No. neurons	150/150
Activation function	ReLU
Loss function	MSE
Optimization	Adam
Learning rate	0.005
Learning rate decay	0.99
Max #epochs	150
Regularization	Early stop (20)
Data processing	Standardization

Table 4.11: Neural network hyperparameters for the prediction of the strain energy

The general results are shown in Figures 4.15 and 4.16 and the general performance measures are shown in Table 4.12. It was decided not to show the average relative error as it does not provide relevant information on the performance. The relative error is high due to the beginning of the target curves being close to zero, 10^{-5} or lower. For example, an error of 0.01, which is accurate considering the range of values of the strain energy, leads to a very high relative error for the values close to zero. Hence the figure shows the absolute error instead. The distribution between targets and predictions in the right bottom figure shows that in general, the distribution resembles the $y = x'$-graph. It can be observed, though, that there is one sequence of points showing a larger deviation. This sequence consists of the predictions and targets from the worst prediction. The best and the worst prediction show that the network has learned the general shape of the strain energy over time. The large maximum

absolute error of 335 for the worst prediction is caused by the steep gradient after the maximum and the prediction reaching the maximum a few days later. The R^2-score of 0.9797 is close to one, which shows that the model has a good fit. The aRRMSE is 0.1219 is below 0.2 and therefore shows that the fit of the network is good. Figure 4.16 shows the predictions and targets of nine samples. These samples show that the predictions can predict the order of the strain energy well, giving accurate predictions of samples with a maximum of 600 and a maximum of 45. It is observed that the predictions of smaller strain energies are less smooth, though absolute differences are small.

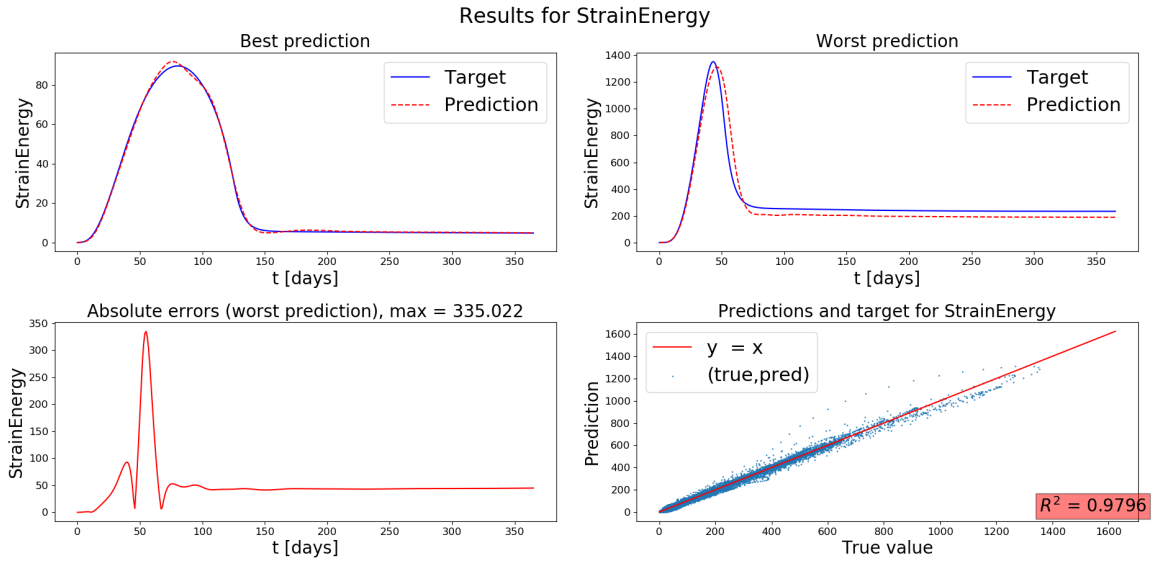


Figure 4.15: Results of the strain energy predictions for one of the validation sets. Upper two graphs show predictions and targets for the best and the worst prediction in terms of MSE. Left bottom graph shows the absolute error for the worst prediction and right bottom graph shows the relation between the predictions and targets, the line 'y = x' for comparison and the R^2-score.

Performance measure	Value
R^2	0.9797 ± 0.0019
aL_2 norm	76.32 ± 5.20
aRRMSE	0.1219 ± 0.0078
Training time	98 s
Validation time	0.0007 s

Table 4.12: General performance for the strain energy predictions.

For the strain energy, the maximum strain energy and the final strain energy are important values and hence the performance for their predictions is considered in more detail in Table 4.13 and Figure 4.17. Figure 4.17a shows the target maxima ranked from smallest to largest and their predictions for all samples in one of the validation sets. In the figure, it can be observed that the maximum values below 400 are more common than above. It is also observed that for higher maxima the prediction varies slightly more. The network reports an MAE of 6.45 which is 0.5% of the full range of maximum values and 3% of the average value. This shows that the network can distinguish well between samples with high and low maximum values. For the final strain energy the same holds, where the network can accurately predict the final values, though it has more difficulty predicting the higher values as they are less common. The MAE of the final strain energy is approximately 0.4% of the range of values and 7% of the average final value. These observations combined with the high R^2-scores above 0.99 for the maximum and final strain energy show that the network can accurately predict the maximum and final strain energy of the samples.

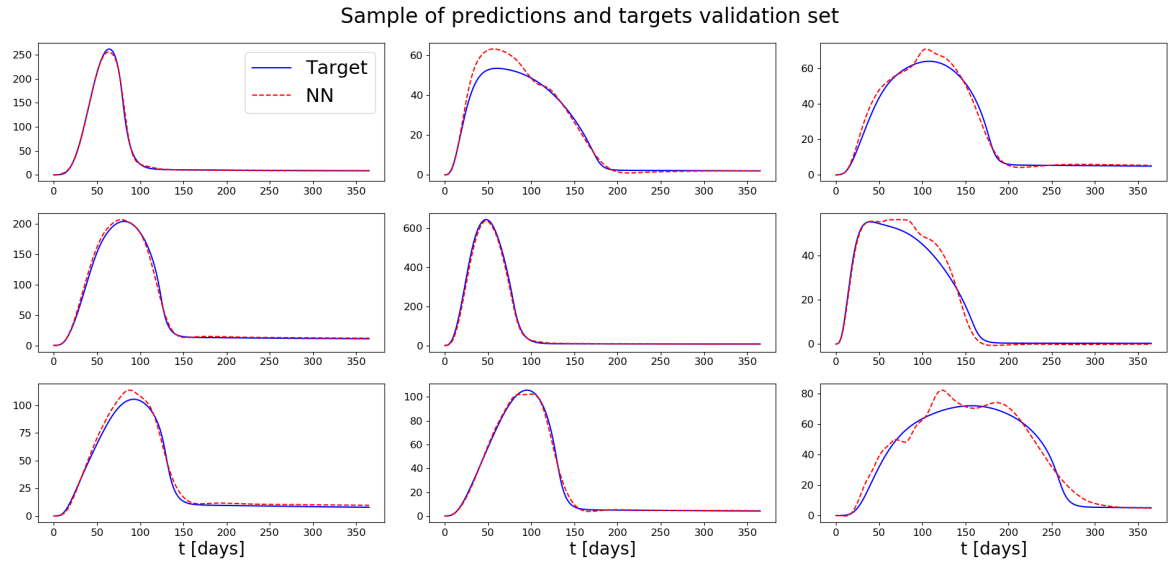
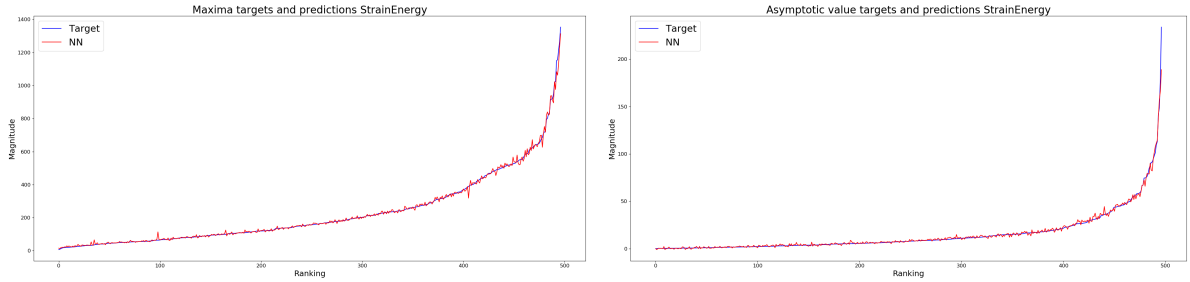


Figure 4.16: Random chosen strain energy samples from the validation set and the corresponding predictions from the neural network.



(a) Maximum strain energy

(b) Final strain energy

Figure 4.17: Distribution of strain energy maxima and final values ranked on the size of the true values.

	MAE	R^2	Min	Max	Range	Average
Max strain energy	6.45 ± 0.41	0.9966 ± 0.001	5.59	1353.19	1347.60	217.2
Final strain energy	1.01 ± 0.11	0.9911 ± 0.002	0.042	233.8	233	13.92

Table 4.13: Results for the maximum and final value of the strain energy averaged over the 10 cross-validation trials. Distribution of the values is provided for context of the Mean Absolute Error.

4.3.2. Test

To the end of finding the final performance of the network, the neural network is trained on both training and validation data and tested on the independent test set. We only report the performance measures as the figures are similar to the performance of the validation set. The general performance measures are similar to the performance of the validation set are close to or within one standard deviation from the mean performance measures for the cross-validation sets. The R^2-score is 0.981 and the aRRMSE of 0.113 show that the fit of the network is good. The low values of the MAE and the high R^2-scores for the maximum and final strain energy with respect to their ranges show that the network can provide accurate predictions. From the performance measures, we conclude that the network has been trained well and can generalize to unseen samples.

4.3.3. Combined predictions

The previous networks have been trained to predict RSAW or strain energy separately. However, the two distributions are similar in the sense that the minimum RSAW is located at approximately the same

Performance measure	Value
R^2	0.981
aL$_2$ norm	70.69
aRRMSE	0.113
Validation time	0.007s

Table 4.14: Performance for strain energy predictions on the test set.

	MAE	R^2	Min	Max	Range	Average
Max strain energy	5.68	0.9973	6.96	1362	1355	220.4
Final strain energy	0.89	0.9940	0.046	149.29	149.25	13.82

Table 4.15: Results for the maximum and final value of the strain energy on the test set. Distribution of the values is provided for context of the Mean Absolute Error.

day as the maximum strain energy. Figure 4.18 illustrates this relation for one sample of the training set. The process over time shows many similarities which can be explained by the direct relation between contraction and strain in the skin. We investigate whether utilizing this relation, by predicting both RSAW and strain energy simultaneously, can further improve the prediction of the strain energy. Training one network to predict both outputs, can also save training time and memory requirements as only one network needs to be trained and saved. Two different approaches are studied for simultaneous prediction. In the first approach the two distributions are transformed by one PCA-transform to combine the information directly. The second approach transforms both distributions separately and uses a parameter-sharing network, which is explained in Section 3.2.3, to utilize the relation.

For the first approach, the values for RSAW and strain energy are concatenated and transformed with a 40-dimensional PCA-transform which can capture the variations in both distributions. A two-layer MLP with 100 neurons in each layer is trained to predict the transformed targets. For the second approach RSAW and strain energy are reduced using a 22- and 25-dimensional PCA-transform, respectively. A parameter-sharing network with one shared layer of 100 neurons and one private layer of 100 neurons for each is used to predict the PCA-targets. The use of shared and private parameters allows the network to learn both inter-target relations and target specific relations. The results from both networks are compared with the individual predictions from the previous sections. In Table 4.16 the performance measures from the predictions of previous sections, the performance of the combined predictions, and the shared-network predictions can be found.

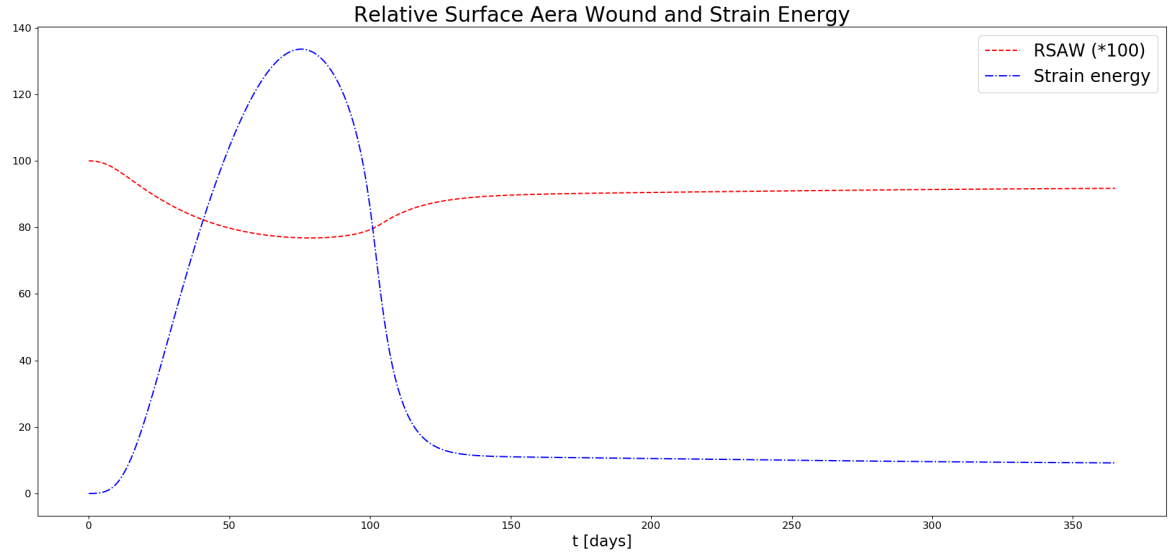


Figure 4.18: Strain energy and RSAW for one sample of the training set. RSAW is scaled by factor 100 for visualization purposes.

Measure	RSAW	RSAW (comb.)	RSAW (sh.)	E_{strain}	E_{strain} (comb.)	E_{strain} (sh.)
R^2	0.9930	0.8723	0.9926	0.9797	0.9771	0.9807
aL$_2$ norm	0.0804	0.499	0.0834	76.32	83.57	74.36
aRRMSE	0.0569	0.3501	0.0601	0.1219	0.1313	0.1175
aRelErr	0.43 %	3.06 %	0.46%	-	-	-
MAE min/max	0.0046	0.0322	0.0048	6.45	7.04	6.24
MAE asymp	0.0025	0.0211	0.0027	1.01	1.088	0.9566

Table 4.16: Performance of the combined predictions for RSAW and strain energy E_{strain} using a neural network to predict full combination of RSAW and strain energy (comb) and one using shared and non-shared layers (sh.). Performance measures are based on the validation set. For ease of comparisons the results from Section 4.2.2 and 4.3.1 are repeated.

The combined approach using the 40-dimensional PCA-transform reports a R^2-score for RSAW of 0.8723 which is significantly lower than the score 0.9939 obtained in Section 4.2.2. The other performance measures for RSAW are significantly worse as well, being 6-7 times as high for the combined approach as for the individual prediction. The prediction of the strain energy on the other hand is relatively similar to the individual prediction in Section 4.3.1, reporting only slightly lower performance values. Based on these observations, it is concluded that by combining the two distributions with one PCA-transform, important information is lost on the characteristics of RSAW. For the parameter-sharing network, the performance for the RSAW is slightly worse and the performance for E_{strain} is slightly higher. However, these differences are within the variation of the 10 cross-validation sets and not significant. Therefore, we conclude that the performance for the shared approach is similar to the separate approaches and both can be trained together, though it does not lead to improvements on the performance.

4.3.4. Strain energy from mechanical values

In this section, a similar approach to Section 4.2.6 is used, where the RSAW was computed using a neural network to predict the displacement. Similarly, the strain energy is computed as a post-processing step from the effective strain ϵ and the concentration collagen ρ computed by the morphoelastic model using Equation (4.2).

$$E_{\text{strain}}(t) = \int_0^{10} E\sqrt{\rho(x,t)}\epsilon(x,t)^2 dx. \tag{4.2}$$

Neural networks are trained to predict the strain ϵ and the concentration collagen ρ separately. To this end, the dimension of the data is reduced using a 55-dimensional PCA-transform for ρ and a 35-dimensional transform for ϵ. The choice for the number of components is based on the explained variance ratio's and a study of the effects on the samples, as is explained in Section 4.2.2. A two-layer network with 200 nodes in each layer is trained using the Adam optimizer with a learning rate of 0.006 with decay factor 0.99 every epoch. Table 4.17 shows the general performance results for the networks for ϵ and ρ in the second and third columns. The R^2-scores for both ρ and ϵ are close to one and the aRRMSE values are below 0.1, which shows the network has learned the distributions well. From both R^2-score and the aRRMSE it can be seen that the predictions of ϵ are more accurate than ρ. Figures 4.19 and 4.20 show the best and worst prediction for ρ and ϵ in terms of MSE at days 5, 25, 50, 150, and 365. For ϵ both the best and worst predictions are close to the target values. For ρ the best prediction is very accurate as well, although the worst prediction shows oscillations especially at days 25 and 50.

In Figure 4.21, nine random samples are shown with both prediction of the strain energy by the network in Section 4.3.1, which we call E^{NN}_{strain}, and the prediction using neural networks to predict the strain and concentration collagen, which we call $E^{\rho,\epsilon}_{\text{strain}}$. To study if the similarity of the prediction can give information on the error, the R^2-score of the two predictions is shown. The performance measures for both predictions are shown in columns four and five of Table 4.17, respectively. The performance of $E^{\rho,\epsilon}_{\text{strain}}$ is slightly worse than E^{NN}_{strain} for all measures, though the R^2-score of 0.9729, the aRRMSE of 0.01183, and the MAE of the maximum and the final contraction of 7.62 and 0.56 indicate that the network does give accurate predictions. Since the networks for ϵ and ρ have been tuned less extensively, it can not be concluded that the direct approach is significantly better. Next, it is studied whether there is a relation between the R^2-score of both predictions and their accuracy. The samples in the third row of Figure 4.21 show predictions that are almost indistinguishable from the target, the R^2-scores above 0.99 indicate that indeed the predictions are very accurate. The second sample in the

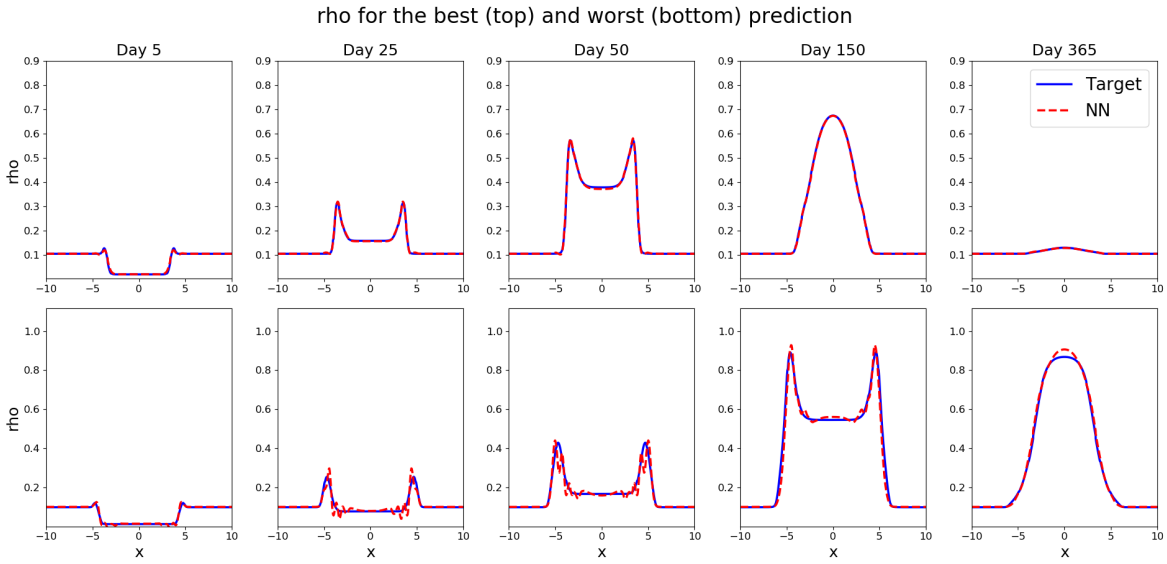


Figure 4.19: Worst and best prediction in terms of MSE for collagen concentration ρ at days 5, 25, 50, 150, and 365.

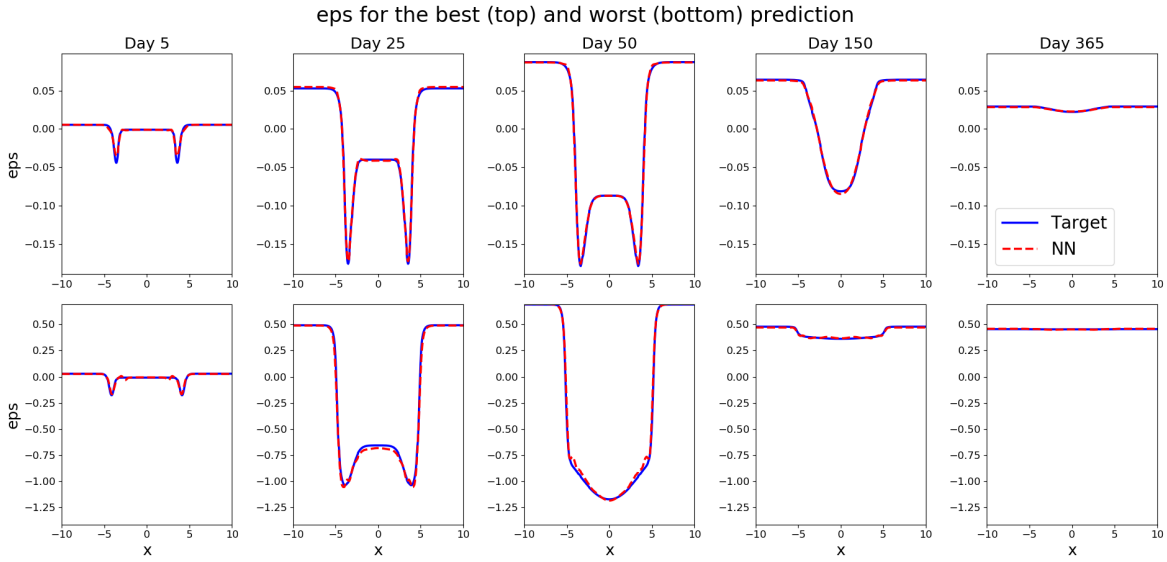


Figure 4.20: Worst and best prediction in terms of MSE for the effective strain ϵ at days 5, 25, 50, 150, and 365.

Performance measure	ρ	ϵ	$E^{\rho,\epsilon}_{\text{strain}}$	E^{NN}_{strain}	Average E_{strain}
R^2	0.9892	0.9939	0.9729	0.9811	0.9847
aL$_2$ norm	1.62	1.11	86.90	78.33	63.92
aRRMSE	0.0872	0.0553	0.1183	0.1178	0.0957
MAE max	-	-	7.62	6.41	5.32
MAE final	-	-	0.56	0.76	0.63

Table 4.17: General performance for the prediction of ρ and ϵ, the prediction of the strain energy using the prediction of ρ, ϵ ($E^{\rho,\epsilon}_{\text{strain}}$), the prediction of the strain energy directly by the NN (E^{NN}_{strain}), and the averaged prediction on the test set.

first column and the first sample in the third column have lower scores of 0.9169 and 0.9177 respectively and for these samples it can be seen that the two predictions vary more in shape. However, it should be noted that these two samples have low maximum values and therefore a lower MSE and MAE. From this it can be concluded that the R^2 score can give an indication of how well the prediction approximates

the shape of the strain energy, though, no conclusions can be drawn on the actual error of the prediction using the R^2-score. From Figure 4.21 it is also observed that although both methods for prediction are

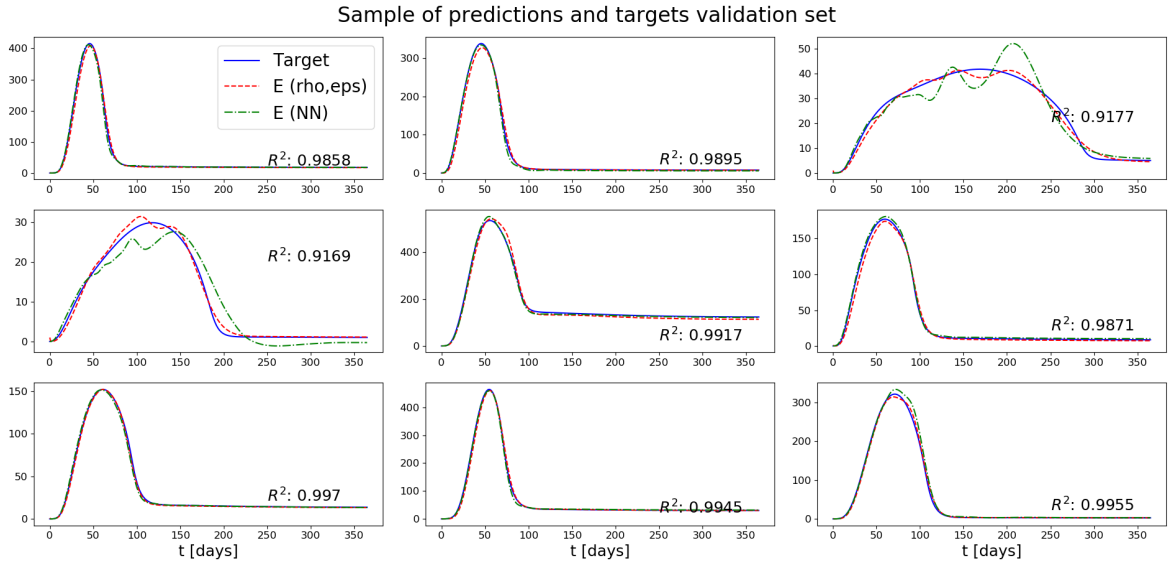


Figure 4.21: Random samples from the test set with the prediction of the strain energy by the neural network directly and by using a neural network to predict ρ and ϵ. The R^2-score between the two predictions is shown as well.

relatively accurate, in some cases they are very similar and in some cases one prediction overestimates where the other underestimates. It is interesting to find if the information from both predictions combined can lead to a better prediction. To that end, the two networks are combined in an ensemble, which provides a new prediction based on the average of the two individual predictions. The results of this prediction are shown in the last column of 4.17. We find that except for the MAE of the final strain energy all performance measures improve when combining the two predictions. We are interested to

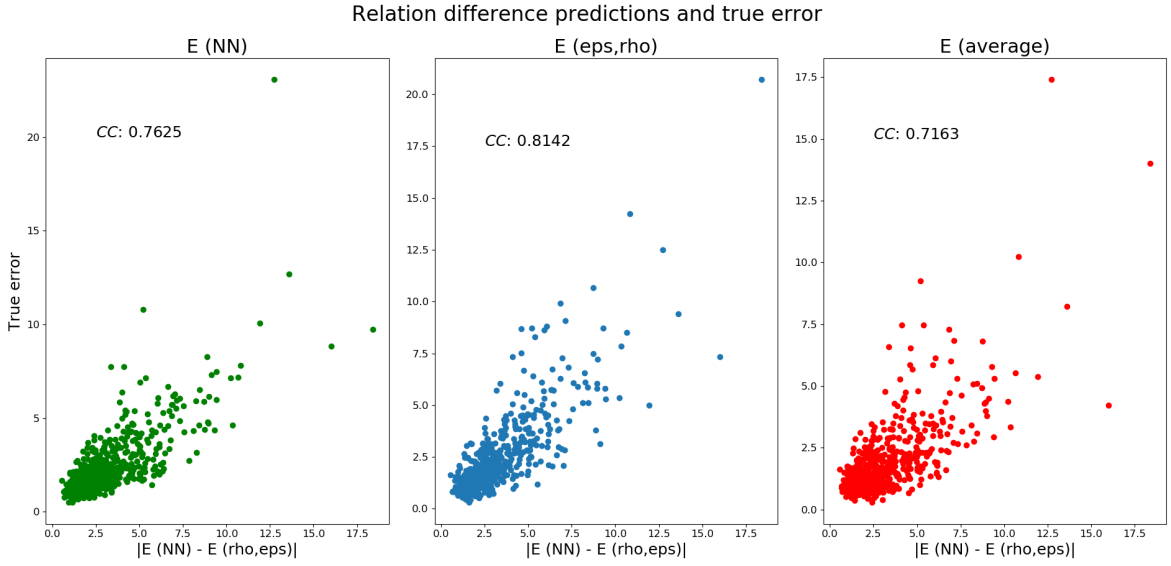


Figure 4.22: Relation between mean absolute difference of the predictions and true mean absolute errors for the predictions $E^{NN}_{\text{strain}}, E^{\rho,\epsilon}_{\text{strain}}$, and for the average prediction. Results are for all samples in the test set. The Pearson correlation coefficient is shown to give an indication of linearity.

find if the similarity of the two separate predictions can give an indication of the true error. To that end, we compute the mean absolute difference between the two predictions E^{NN}_{strain} and $E^{\rho,\epsilon}_{\text{strain}}$ and the

mean absolute errors with respect to the targets. Figure 4.22 shows the distribution with the mean absolute difference on the x-axis and the respective true errors on the y-axis. The Pearson correlation coefficient is shown indicating to what extent the values show a linearity relation. The results indicate a relatively strong correlation, hence the mean absolute difference between the two predictions can give an indication of the true error of the solution. We note that the correlation for the average prediction is less than for the individual predictions.

From this study, it is concluded that a neural network can be trained to predict the effective strain and the collagen concentration over time and in space accurately. These predictions can be used to provide accurate predictions of the strain energy over time. Furthermore, it is concluded that combining these predictions with the direct surrogate for the strain energy in an ensemble can lead to better predictions. It must be noted that it can not be concluded that the ensemble prediction is always better.

4.4. Two-dimensional morphoelastic model

From the results for the one-dimensional model in the previous sections, it is evident that neural networks can predict the outcomes accurately and can be used as a computationally cheap alternative. However, a one-dimensional model has large restrictions when modeling a three-dimensional wound and higher dimensional models are currently being developed to increase the accuracy of the simulations. In this section, the two-dimensional morphoelastic model is used. This model is implemented by Barion [12] and elaborates on the morphoelastic framework developed by Koppenol [50]. The addition of a second dimension increases the number of grid points and the size of the linear systems leading to a large increase in the computation time. The application of a neural network surrogate can prove to be especially useful for these expensive simulations. A new dataset needs to be generated for the two-dimensional model, which is discussed in Section 4.4.1. In Sections 4.4.2 - 4.4.4 surrogate are discussed for the relative surface area of the wound, the strain energy, and the movement of the wound edge.

4.4.1. Dataset

For the generation of the dataset for the two-dimensional morphoelastic model, the same approach is used as for the one-dimensional case in Section 4.1. For each of the input parameters that vary between simulations, a suitable range of values is defined. At the start of each simulation, all the input parameters are drawn uniformly from their respective range. Most of the input parameters are used in both models. However, the two-dimensional model has an additional parameter ν which accounts for the Poisson effect. This means that the range for E needs to change as it implicitly incorporated the Poisson effect for the one-dimensional model. In two dimensions the parameter μ can vary between the directions and μ_1 and μ_2 are used for x- and y-direction, respectively. We have assumed that there is no difference in viscosity between x- and y-direction, i.e. $\mu_1 = \mu_2$.

Even though most parameters are the same for both models, for some parameters the ranges needed to be adapted to prevent unstable or unrealistic results. It was observed that the two-dimensional model is especially sensitive to the values of $D_F, r_F^{\max}, k_F, \bar{N}, \delta_N, \delta_M$ and k_c, causing a larger variation in the relative surface area than changes in the other parameters. It must be noted that only individual parameter sensitivity has been studied and no interaction between parameters has been considered. The ranges of the sensitive parameters were adapted to prevent instabilities and are highlighted in bold in Table 4.18. It has to be noted that a thorough study of the parameter and model sensitivity and their interaction is necessary to improve the ranges for future studies. The sensitivity results for the parameters obtained by Egberts et al. [22] can be extended to the two-dimensional model.

Parameters that are fixed for all patients are given in Table 4.19 for reproducibility reasons. Note that L is a fixed parameter for the two-dimensional model. Explanation and references for the parameter ranges can be found in Appendix A. Since the simulations of the two-dimensional model are computationally more expensive, fewer simulations are run and each simulation is computed until 100 days instead of 365 days. For the simulations, the results are computed on the quarter of the domain, i.e. $\Omega = [0, 4] \times [0, 4]$, where the wound is defined by the domain $\Omega^w = [0, 1] \times [0, 1]$. The domain is discretized using uniform bilinear elements generated by a grid of 20x20 grid points with a time step of 0.1 day. As discussed in Chapter 2, the two-dimensional model computes four constituents (N, M, ρ, c), five mechanical values $(v_1, v_2, \epsilon_{11}, \epsilon_{22}, \epsilon_{33})$, the relative surface area (RSAW), and the strain energy (E_{strain}). Furthermore, the dataset includes the movement of the edge of the wound and the location

Parameter	Range
$\mathbf{D_F}$	$7.6167 \cdot 10^{-7}$ - $1.2 \cdot 10^{-6}$
χ_F	$(2-3) \times 10^{-3}$
D_c	$(2.22-3.2) \times 10^{-3}$
r_F	$0.832-0.924$
$\mathbf{r_F^{max}}$	$2-2.3$
$\mathbf{k_F}$	$8 \times 10^6 - 1.08 \times 10^7$
$\mathbf{\bar{N}}$	$(1-1.5) \times 10^4$
$\bar{\rho}$	$0.0975-0.1200$
$\delta_{\mathbf{N}}$	$0.0119-0.02$
$\delta_{\mathbf{M}}$	$0.055-0.065$
δ_c	$(0.354-0.693) \times 10^{-3}$
δ_ρ	$(5.5-6.075) \times 10^{-6}$
$\kappa_{\mathbf{F}}$	$0.5 \cdot 10^{-6} - 1.6 \cdot 10^{-6}$
$\tilde{c}$	$(1-5) \times 10^{-8}$
a_c^I	$(0.9-1.1) \times 10^{-8}$
a_c^{II}	$(0.98-1.02) \times 10^{-8}$
a_c^{III}	$(2-2.5) \times 10^8$
a_c^{IV}	$(0.8-1.2) \times 10^{-9}$
$\mathbf{k_c}$	$(0.8-0.9)\delta_c \bar{\rho} a_c^{II}$
ρ_t	$0.89-1.29$
μ_1	$60-1000$
μ_2	μ_1
$\mathbf{E}$	$28-34$
ξ	$(4.4-4.8) \times 10^{-2}$
ζ	$(4-9) \times 10^2$

Table 4.18: Parameter values which are varied for each patient in the two-dimensional dataset. For the bold parameters, the ranges are changed with respect to the one-dimensional dataset.

Parameter	Value
$k_\rho^{\max}$	10
$\bar{M}$	0
$\bar{c}$	0
q	$\frac{[\log(\delta_N)-\log(r_F(1-\kappa_F\bar{N})]}{\log(\bar{N})}$
η^I	2
η^{II}	0.5
$\tilde{N}$	$0.2\bar{N}$
$\tilde{\rho}$	0
k_ρ	$\delta_\rho \bar{\rho}^2$
R	0.995
ν	0.49
$\mathbf{L}$	1

Table 4.19: Parameter values that are fixed for all patients in the two-dimensional dataset. The bold parameters have been changed with respect to the one-dimensional dataset.

of the grid points over time. The keys and sizes of the values in the dataset are given in Table 4.20. The simulations are computed on a server of 27 nodes, where each node has four or eight cores. The simulations are computed in parallel on the cores of each node. An average simulation needs four hours to finish on a node with four cores.

Key	**Size**
Inputs	$N_{\text{samples}} \times 26$
RSAW	$N_{\text{samples}} \times 100$
StrainEnergy	$N_{\text{samples}} \times 100$
N	$N_{\text{samples}} \times 100 \times 441$
M	$N_{\text{samples}} \times 100 \times 441$
c	$N_{\text{samples}} \times 100 \times 441$
ρ	$N_{\text{samples}} \times 100 \times 441$
v_1	$N_{\text{samples}} \times 100 \times 441$
v_2	$N_{\text{samples}} \times 100 \times 441$
ϵ_{11}	$N_{\text{samples}} \times 100 \times 441$
ϵ_{12}	$N_{\text{samples}} \times 100 \times 441$
ϵ_{22}	$N_{\text{samples}} \times 100 \times 441$
x	$N_{\text{samples}} \times 100 \times 441$
y	$N_{\text{samples}} \times 100 \times 441$
bnd	$N_{\text{samples}} \times 100 \times 21$

Table 4.20: Two-dimensional dataset - keys and sizes.

During the generation of the two-dimensional dataset, the numerical instabilities were not all caught,

resulting in a polluted dataset. In most cases, the numerical instabilities caused exploding strain energy computations and unrealistic behaviour of the wound edge and these samples were filtered from the dataset. It should be noted that in hindsight an implementation error was found in the numerical model, such that some of the simulations might not be accurate. This could have led to more numerical instabilities. The final dataset contains 900 samples and due to this limited number of samples, it has been decided to use 10-fold cross-validation only to validate the performance of the model and to leave out the independent test set.

4.4.2. RSAW prediction

Analogously to the one-dimensional case, a network is trained to predict RSAW over time, using the direct surrogate approach. As the data type and information are similar, the same type of network can be used. In Section 4.2.3, it was found that the predictions of the relative surface area could be improved by adding more data until the training set contained approximately 3200 samples. Since there are only 900 samples in the two-dimensional dataset, the results are expected to be less accurate than for the one-dimensional dataset. The chosen network hyperparameters can be found in Table 4.21.

Type NN	MLP
No. neurons	100/100
Activation function	ReLU
Loss function	MSE
Optimization	Adam
Learning rate	0.008
Learning rate decay	0.99
Max #epochs	150
Regularization	Early stop (20)
Data processing	Standardization

Table 4.21: Neural network parameters RSAW prediction for the two-dimensional model.

Figure 4.23 shows the best and worst predictions and the distribution between predictions and targets for one of the cross-validation sets. In the figure, it is observed that the best prediction is accurate and closely resembles the target. For the worst sample, retraction starts too early and a maximum relative error of 6.26% is obtained. The distribution between targets and predictions resembles the $'y = x'$-line and shows that the predictions are close to the target values. Table 4.22 gives the general performance measures for the network. As expected, the performance on the two-dimensional dataset is lower than the performance on the one-dimensional dataset. The R^2-score is 0.9577 compared to 0.9930 and the aRRMSE is 0.167 compared to 0.05 for the one-dimensional dataset. However, the average relative error of 0.74% and the aRRMSE, which is below 0.2, show that the overall performance is still good. Figure 4.24a shows the prediction for nine random samples, from which it can be observed that the neural network predictions are a good approximation of the targets.

Performance measure	Value
R^2	0.9577 ± 0.0062
aL_2 norm	0.070 ± 0.007
aRRMSE	0.167 ± 0.0179
aRelErr	$0.74 \pm 0.07\%$
Training time	18 s
Validation time	0.0006 s

Table 4.22: General performance for RSAW predictions on the two-dimensional dataset.

	MAE	R^2	Min	Max	Range	Average
Min RSAW	0.0078 ± 0.0007	0.979 ± 0.005	0.532	0.933	0.401	0.753

Table 4.23: Results for the minimum RSAW values averaged over the 10 cross-validation trials for the two-dimensional dataset. Distribution of the values is provided for context of the Mean Absolute Error.

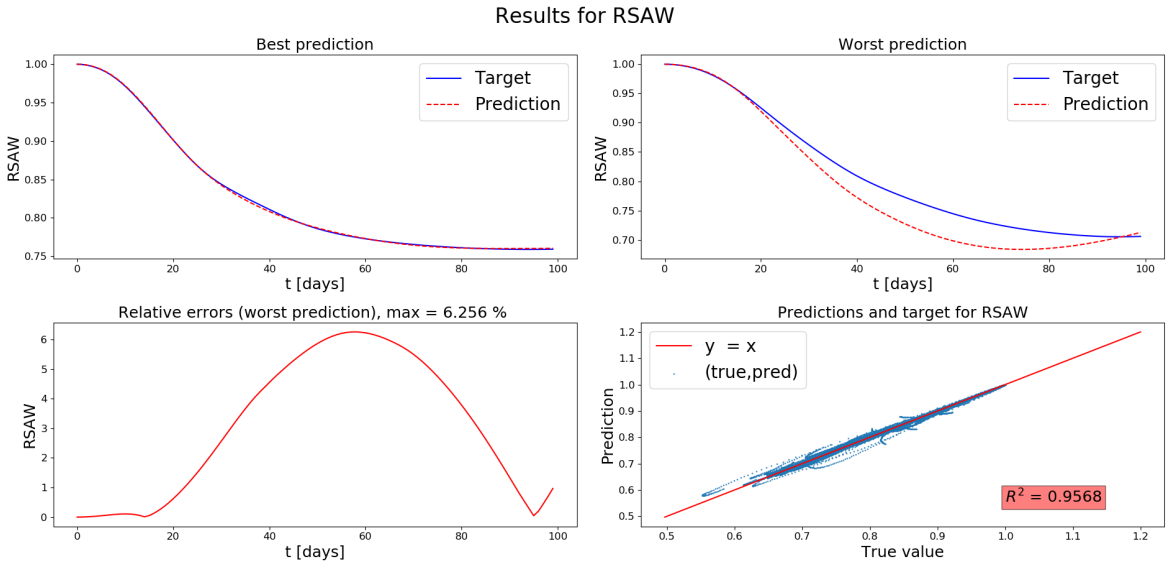
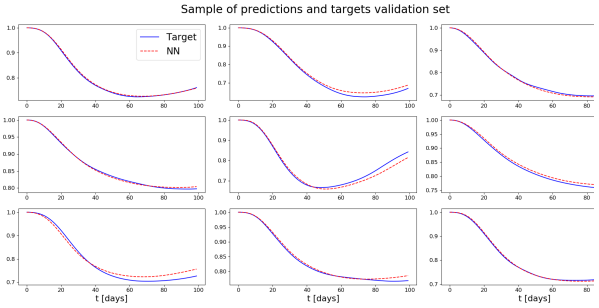
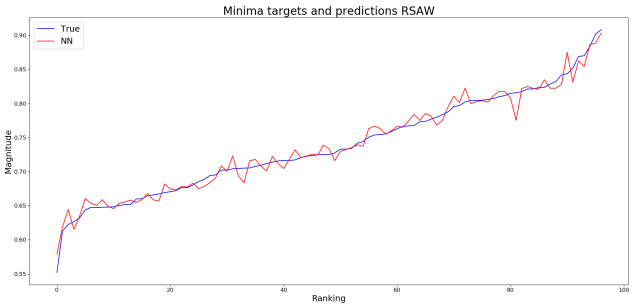


Figure 4.23: Results RSAW prediction for the two-dimensional model. Upper two graphs show predictions and targets for the best and the worst prediction in terms of MSE. Left bottom graph shows the relative error for the worst prediction and right bottom graph shows the relation between predictions and targets, the line 'y = x' for comparison and the R^2-score.



(a) Targets and predictions for nine random chosen samples of the validation set.



(b) Minimum RSAW

Figure 4.24: Results for the RSAW prediction of the two-dimensional model on one of the cross-validation sets.

Since the simulations are run for 100 days only, the process is still in the retraction phase and there is no final contraction. In most cases, the minimum RSAW has been reached and the results for the minima can be found in Figure 4.24b and Table 4.23. The predictions of the maximum contraction are accurate with a mean absolute error which is 1.9% of the range of the values. This shows that the neural network can distinguish well between the magnitudes of the different samples, which is also visualized in Figure 4.24b. From Table 4.23, it is noted that the range of values is smaller than for the one-dimensional model, showing less variation in the maximum contraction. The lower variation can be caused by the adapted ranges for the parameters and by the smaller dataset. From the observations, it is concluded that the network can learn the distribution quite well and that it is expected that the results will improve if more training samples are added.

4.4.3. Strain energy prediction

Similarly to the one-dimensional dataset, a surrogate approach for the strain energy is studied for the two-dimensional model as well. The strain energy is computed as described in Section 2.2.3. A two-layer MLP network is used with 100 nodes in each layer and ReLU activation units. The network is trained with the Adam optimizer using a learning rate of 0.008 with an exponential learning rate decay factor 0.99 every epoch. The network is trained for at most 150 epochs.

The general results are shown in Table 4.24 and Figure 4.25. Analogously to the one-dimensional

case, the absolute error is shown instead of the relative error for the strain energy. Both the higher aRRMSE value of 0.3586 and the lower R^2-score of 0.8426 indicate that the neural network prediction is significantly worse than for the one-dimensional predictions. Furthermore, the two right figures in Figure 4.25 show that the neural network has difficulty predicting the right shape. The distribution of predictions and targets shows larger deviations from the $'y = x'$-line, indicating the fit is less accurate. Figure 4.26a shows the predictions and targets of nine random samples. For these nine samples, the predictions are fairly close to the targets.

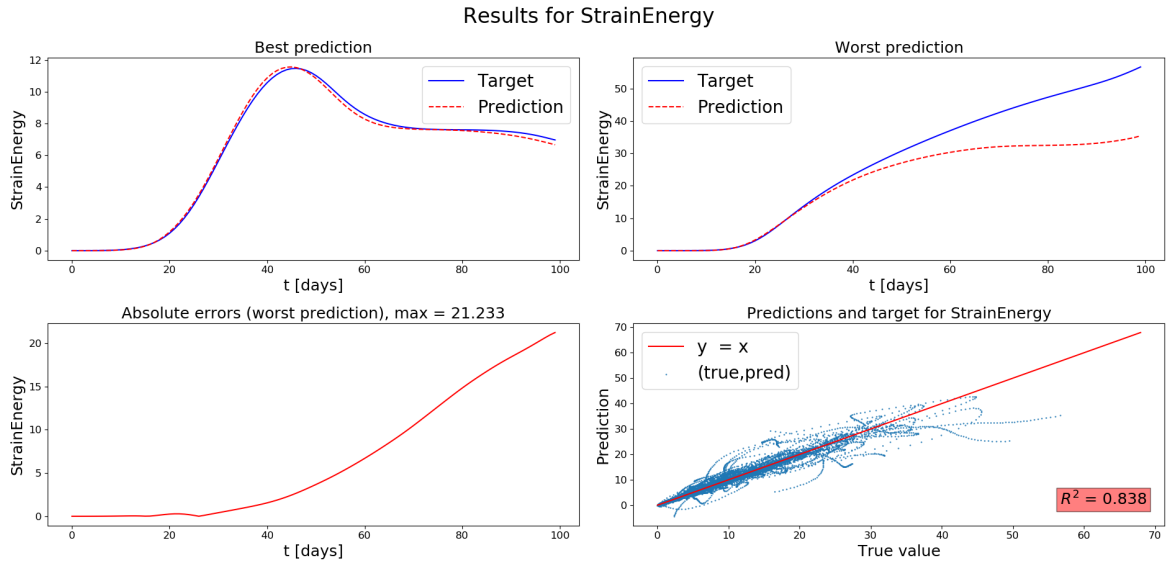
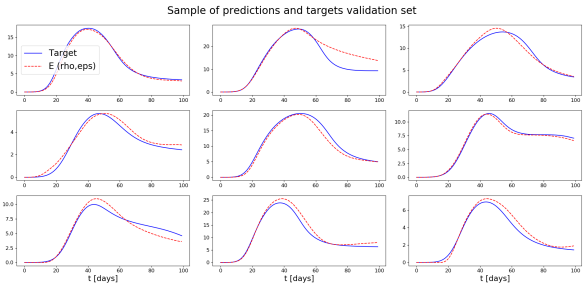


Figure 4.25: Results strain energy prediction for the two-dimensional model. Upper two graphs show predictions and targets for the best and the worst prediction in terms of MSE. Left bottom graph shows the absolute error for the worst prediction and right bottom graph shows the relation between the prediction and targets, the line 'y = x' for comparison and the R^2-score.

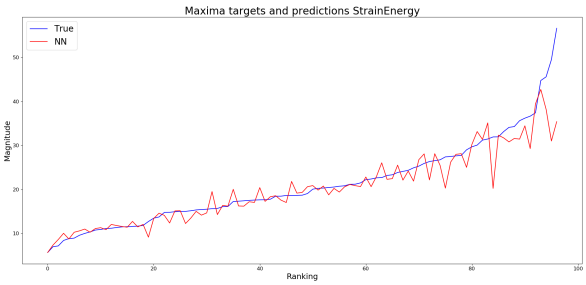
Performance measure	Value
R^2	0.8426 ± 0.0231
aL_2 norm	22.24 ± 3.10
aRRMSE	0.3586 ± 0.0299
Average training time	15 s
Validation time	0.005 s

Table 4.24: General performance for strain energy predictions on the two-dimensional dataset over the 10 cross-validation trials.

Similarly to the relative surface area, the strain energy has not reached a final state in all the samples and only the performance on the maximum strain energy can be considered. The performance on the maximum strain energy is reported in Table 4.25 and Figure 4.26b. In the figure, it can be seen that the predictions of samples with a higher maximum are less accurate than for samples with a lower maximum. From Table 4.25 it is noted that the range of values for the strain energy is much lower than for the one-dimensional model where the maximum strain energy could reach values over 1000. The smaller range is likely caused by the more limited ranges for sensitive parameters and the smaller dataset. The mean absolute error of the maximum strain energy is below 5% of the range of values which shows that a reasonable indication of the maximum intensity of the strain energy can be given. Though, from Figure 4.26b it can be observed that the network has more difficulty predicting higher maxima. From the observations, it is concluded that the overall performance of the network is not very good and has difficulty generalizing to samples with a higher minimum. For the samples with a lower minimum, the network can give reasonable predictions. More samples should be added to the training set to find whether the performance can be improved.



(a) Random chosen samples from the validation set and the corresponding predictions from the neural network.



(b) True values and predictions for the maximum strain energy for all samples in the validation set.

Figure 4.26: Results for the strain energy predictions of the two-dimensional model.

	MAE	R^2	Min	Max	Range	Average
Max strain energy	2.37 ± 0.49	0.832 ± 0.0767	2.67	80.61	77.94	21.35

Table 4.25: Results for the maximum strain energy values averaged over the 10 cross-validation trials. Distribution of the values is provided for context of the Mean Absolute Error.

4.4.4. Wound edge prediction

In two dimensions the geometry of the wound becomes more important and for the two-dimensional model the relative surface area only gives information about general contraction and not about localized contractions. Using the displacement of the wound edge, it can also be shown how the wound contracts and retracts. Furthermore, a visualization of the wound movement is intuitively easier to interpret than numeric values. In this section a neural network is trained to predict the movement of the wound edge over time. We note that the initial wound size and shape are equal for all samples in the dataset.

The dimension of the boundary outputs is reduced using a 20-dimensional PCA-transform and standardized to improve the training. A two-layer network is trained with 40 neurons in each layer. The network is trained for 150 epochs with learning rate 0.007 and learning rate decay factor 0.99. For visualization purposes, the target and the prediction are displayed at day 0, 25, 50, 75, and 99. These days are shown for the best and the worst prediction in terms of the L_2-norm in Figure 4.27. From the timestamps in the figure, the contraction and start of retraction of the scar can be seen. For the best sample, the prediction is very accurate. For the worst sample, the prediction is very accurate until 50 days, after which a slight deviation occurs. The performance measures are reported in Table 4.26. Here the average relative error is only considered on boundary points unequal to zero on both axes, such that the results remain valid. The R^2-score of 0.9614 and the aRRMSE of 0.19 indicate a fairly good fit of the model. From the observations, it is concluded that the network can provide fairly accurate predictions of the movement of the wound edge. Similarly to the prediction of RSAW and the strain energy, the performance of the model will likely increase when it is trained on more samples. Therefore, the neural network can be used to provide visualizations of the wound contraction.

Performance measure	Value
R^2	0.9614 ± 0.0054
aL_2 norm	0.1751 ± 0.0171
aRRMSE	0.1901 ± 0.0185
aRelErr*	$3.78 \pm 0.04\%$
Average training time	63 s
Validation time	0.0006 s

Table 4.26: General performance for wound edge prediction on the two-dimensional dataset. *Average relative error taken over all values unequal to zero on both axis.

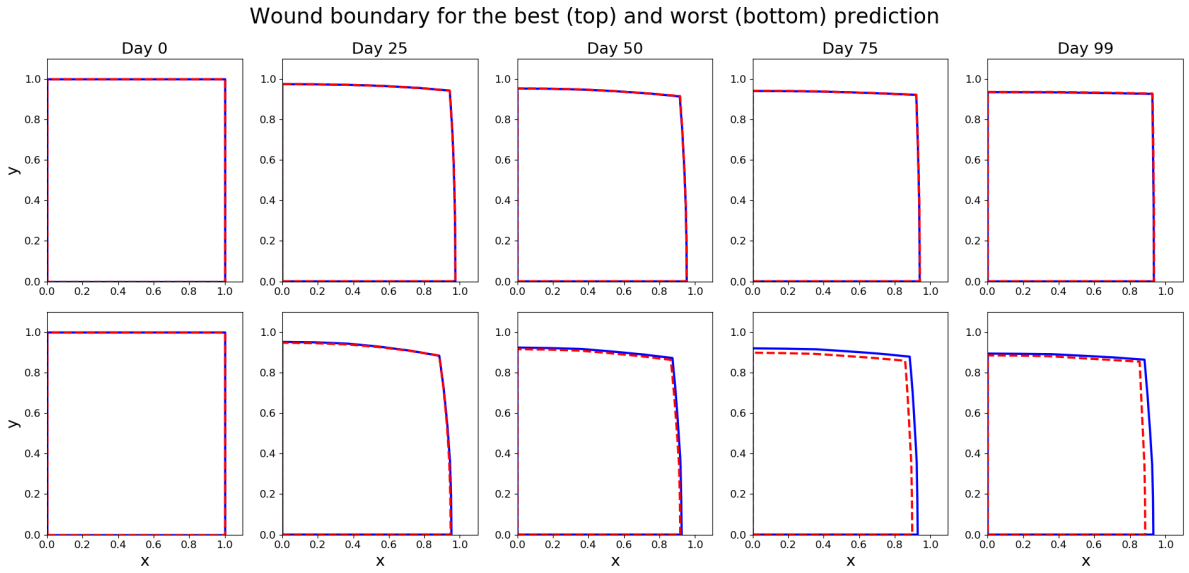


Figure 4.27: Worst and best prediction of the validation set for the wound edge movement shown at days 0, 25, 50, 75 and 99. The blue line shows the true movement and the red dashed line the neural network prediction.

4.5. Conclusion

In this chapter, we studied the use of neural network surrogates for both the one- and two-dimensional morphoelastic models. We conclude that neural network surrogates can be trained to predict the RSAW and the strain energy for the one-dimensional model very accurately. Furthermore, it is shown that the surrogate approach achieves a large acceleration in computation time as it can predict all simulations in the test set (1222 samples) in approximately 0.007 seconds, where the numerical model needs approximately 90 seconds to compute one simulation. It is shown that the PCA-transform can be valuable in reducing dimension without losing the essential information for training the network. It is found that an explained-variance weighted loss function is necessary to ensure proper training for the PCA-targets. We considered approaches where the neural networks were trained to predict the displacement of the wound, the collagen concentration, and the effective strain. From these predicted constituents and mechanical values, the RSAW and the strain energy can be derived. It was shown that these methods are effective as well in predicting the RSAW and the strain energy. These methods were combined with the direct surrogate predictions to function as an ensemble, which led to a slight improvement with respect to the individual predictions. Moreover, for the strain energy, it was shown that the mean absolute difference between the two predictions is correlated with the true mean absolute error and hence can provide a means for uncertainty estimation.

The preliminary study of the two-dimensional model showed that the neural networks can provide a significant acceleration as well, although the results are less accurate than for the one-dimensional model. The lower accuracy is likely caused by the smaller training set, 600 vs 4200 samples. The two-dimensional model has the advantage that the movement of the wound edge can be predicted, which can be valuable for visualization purposes. It can be concluded that even for the small training set a neural network can be used to predict this movement fairly accurately. Therefore, it is concluded that a neural network surrogate can be valuable in providing fast predictions for the wound area and the movement of the scar. The prediction time for the two-dimensional model is decreased from four hours for one simulation to 0.008 seconds for 85 simulations. It is noted that increasing the number of samples in the training set is necessary to further improve the performance of the surrogate models.

5

Hybrid Model

In the previous chapters, the neural network is used as a surrogate for the complete morphoelastic model, i.e. to find the prediction for the desired output variable directly from the input parameters. However, this approach limits the user's choices compared to the morphoelastic model. In the morphoelastic model the user can choose, for example, how many days to predict ahead, which type of initial conditions to use, and whether to make adaptations after a certain number of time steps. The neural networks in the surrogate approach were trained to predict a fixed number of days ahead. To maintain the flexibility of the morphoelastic model a hybrid approach is developed. In this approach, only the computationally expensive steps of the morphoelastic model are replaced by a neural network. The computationally expensive task is the computation of the constituents and mechanical values for the discretized time step. This computation can be seen as a nonlinear mapping of the values at time step t and the input parameters to the values at time step $t + \Delta t$. Hence, it is possible by the universal approximation theorem to approximate this mapping with a neural network [27]. Combining the morphoelastic model and the neural network into a hybrid model would get the fast computations of the neural network with the flexibility of the morphoelastic model.

In this section, a preliminary study is performed on training a neural network to predict the constituents and mechanical values for the next time step based on their values at the current time step. The neural network predictions can be applied iteratively until the required day is reached. Section 5.1 describes the data format and data processing and Section 5.2 discusses the network architecture and the results.

5.1. Dataset

The data, generated for the one-dimensional dataset in Chapter 4, can be used for training the hybrid model as well. More details on the generation of the dataset and the parameter values can be found in Section 4.1 and Appendix A, respectively. Although the data itself can be used, a different format for the data is required. To find the required format, the inputs and outputs of the nonlinear mapping that we want to approximate must be considered. The morphoelastic model computes the new constituents and mechanical values, $M^{t+1}, N^{t+1}, \rho^{t+1}, c^{t+1}, u^{t+1}, v^{t+1}, \epsilon^{t+1}$, from the old constituents and mechanical values. In these computations, the input parameters p are used as well. Therefore, the inputs of the neural network must be $M^t, N^t, \rho^t, c^t, u^t, v^t, \epsilon^t$ and p and the outputs of the neural networks must be $M^{t+1}, N^{t+1}, \rho^{t+1}, c^{t+1}, u^{t+1}, v^{t+1}, \epsilon^{t+1}$. The constituents and mechanical values are all vectors with length 202, equal to the number of grid points in the original simulations. To summarize, a neural network is taught to reproduce a nonlinear mapping $\mathcal{F} : \mathbb{R}^{7\times 202} \times \mathbb{R}^{25} \rightarrow \mathbb{R}^{7\times 202}$:

$$[M^{t+1}, N^{t+1}, \rho^{t+1}, c^{t+1}, u^{t+1}, v^{t+1}, \epsilon^{t+1}] = \mathcal{F}(M^t, N^t, \rho^t, c^t, u^t, v^t, \epsilon^t, p). \tag{5.1}$$

The previously constructed dataset in Section 4.1 contains the values for all time steps per sample. To process the data for training, all samples are split along the time axis. As shown by Grzeszezuk et al. [33] the neural network does not need to satisfy the time step stability criteria of the numerical model and larger time steps can be considered. It is chosen to split the dataset such that each sample predicts five days ahead, which decreases the number of total steps and still ensures that the minimum

RSAW can be captured properly. With this approach, the number of available samples is increased with factor 360 as each original sample in the dataset is now split into 360 separate samples. The constituents and mechanical values at the input time step are combined with the input parameters to form the input for the neural network. The output consists of the constituents and mechanical values five days later. Furthermore, the true RSAW and strain energy distributions over 365 days and the initial conditions ($t = 0$) are saved for means of evaluation.

For training and validation, an 80:20% split of the obtained samples is used. Due to the splitting of each sample into 360 separate samples and the random distribution between training and validation set, these two sets have an overlap in terms of the 25 input parameters. To ensure that the test set contains samples that are different from the training set regarding the 25 input parameters, the test set is constructed from original samples that have not been used for the generation of the training and validation set. This results in a training set of 1.9 million samples, a validation set of 190 000 samples, and a test set of 298 080 samples.

Before training the inputs and targets need to be standardized. The inputs, except for the input parameters p, and the targets are from the same distribution and can be standardized using the same transform. This is an advantage for performing iterative predictions as the prediction at time step t can directly be used as input for the predictions of time step $t + 1$. The input parameters p are standardized using a separate transform. The transformed data is used to train the network.

5.2. Network & Results

In this section the performance measure, the network architecture, and the preliminary results are discussed. Due to limited computational resources and limited time, only a small study could be performed. The following approach is taken to demonstrate the concept and the possibilities of the method. First, a network is trained on 15% of the training data and evaluated on validation and test set. Then the training set size is increased to 30% of all training samples and the network is retrained. The performance of both trained networks on the test set is compared. A positive effect of more training samples on the performance can show, to some extent, whether the performance will improve by increasing the number of samples to the complete training set.

5.2.1. Performance measures

In order to evaluate the performance of the neural networks, two different methods are used. The general performance of the predictions for the constituents and the mechanical values is computed on the validation set using the goodness of fit R^2, the aRRMSE, and the average 2-norm as described in Section 3.7. The performance on the validation set shows how well the network can predict the next time step. It is more interesting to know how well the model is able to predict RSAW and the strain energy in the future. To that end, the inputs of test samples at $t = 0$ are used to initiate 73 consecutive predictions (365 days in total), each using the previous prediction as new input. From the obtained predictions of the constituents and mechanical values for 365 days the RSAW and strain energy can be computed. These predictions are compared to the true RSAW distribution for each sample. The performance is evaluated using the goodness of fit R^2, aRRMSE, and the average L_2-norm.

5.2.2. Network

The constituents and mechanical values inherit the same spatial structure of the mesh in the numerical model. This spatial structure can be utilized by a Convolutional Neural Network (CNN), which is specialized in recognizing spatial structures. Based on this property, it is chosen to train a CNN for the hybrid model. The constituents and mechanical values at the time step t represent the channels in the input for the convolutional layer. Two convolution layers of seven input and seven output channels with kernel size three are used to extract the important features of the constituents and mechanical values. The outputs of the convolutional layers are passed to a MaxPool layer with kernel two and stride two, to reduce the dimension. The input parameters p do not have a spatial structure and cannot be passed to the convolution layers. Therefore, these inputs are concatenated to the outputs of the MaxPool layer. The combined values are passed through two linear layers with ReLU activations of 250 neurons, and an output layer of size 7*202. The outputs from the output layer represent the constituents and mechanical values at the next time step. The network is trained using the Adam optimizer with learning rate 0.0005 and is trained for only 15 epochs, due to the time limitations.

5.2.3. Results 15% training data

First, the CNN is trained on 15% of the training data, i.e. approximately 228 000 samples. Although the number of samples is significantly higher than for the surrogate networks, this dataset contains approximately 720 input parameter combinations from the original dataset. The results of a 200 sample batch of the validation set are shown in Table 5.1 for the constituents. The performance on the validation set is for the predictions of the next time step for all constituents and mechanical values. It is noted that the results on the validation set are for a batch of 200 samples only and therefore are not necessarily the performance on the complete validation set. These values are only used as an indication of how well the network is trained, the general performance of the hybrid approach is determined based on the predictions on the test set. For the given batch, the performance is good with R^2-scores close to one and low aRRMSE scores, which are below 0.1 except for v. The aL$_2$-norms are dependent on the magnitude of the values and therefore it is expected that the values for M and N are higher.

Performance	M	N	c	ρ	u	v	ϵ
R^2	0.9916	0.9981	-	0.9969	0.9889	0.9752	0.9980
aRRMSE	0.0547	0.0423	-	0.0525	0.0336	0.1211	0.0403
aL^2	1057	2800	-	0.1045	0.1195	0.0081	0.0561

Table 5.1: Performance measures for the constituents and mechanical values on a 200 sample batch of the validation set, using a CNN network trained on 15% of the training data. Due to an error the values of c were not saved correctly.

It is more interesting to find the results on the test set. For 240 samples in the test set the initial conditions, i.e. the constituents and mechanical values at $t = 0$, are passed to the neural network. Iteratively, the predictions for days 0 to 365 with a time step of five days are computed. The predictions for the constituents and mechanical values are used to predict the relative surface area and the strain energy over time. Figure 5.1 shows the best and the worst prediction for u, ρ, ϵ in the test set at days 5, 50, 100, 250, and 365. Note that the network also predicts the values of M, N, c, and v, which are not shown here as they are not needed for the computation of RSAW and the strain energy. It is observed that the best prediction is close to the targets for ρ and ϵ and u, showing some deviation on maximum and minimum values only. For the worst sample, it is observed that the predictions for u and ϵ are too low and seem to converge too fast to a final value. The prediction for ρ shows a different distribution over space for day 50 and 100. From these observations, it is clear that in the best case the predictions are fairly accurate and in the worst case the predictions are wrong.

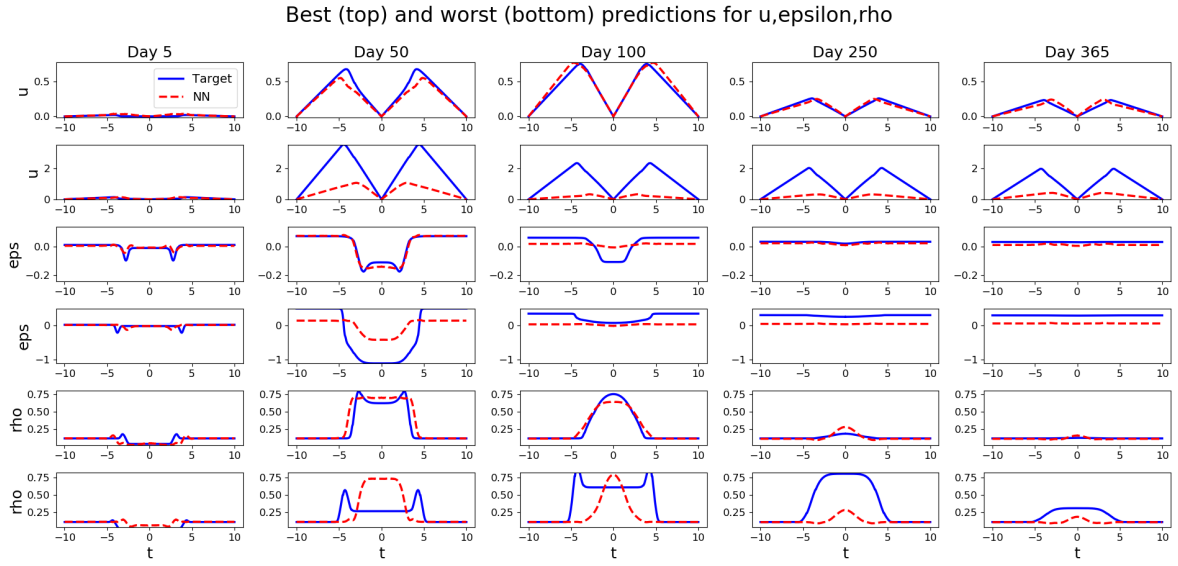
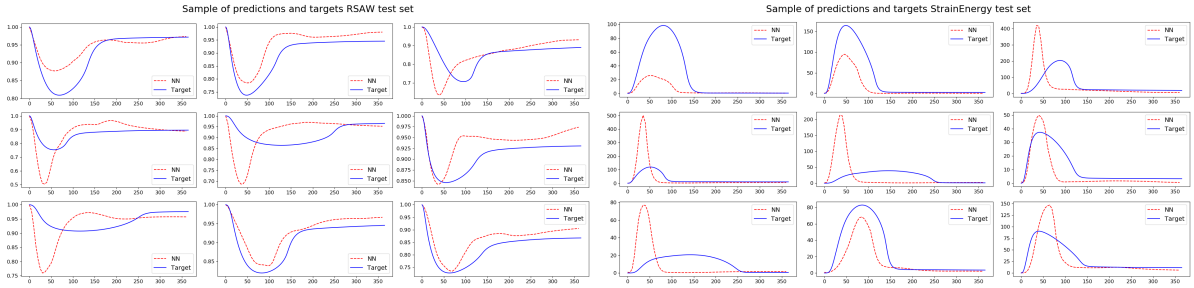


Figure 5.1: Best and worst predictions in terms of MSE for the displacement of the wound u, the concentration of collagen ρ and the effective strain ϵ at days 5, 50, 100, 250, and 365 for the test set. Predictions are computed by the CNN trained on 15% of the training data.

The distributions of RSAW and strain energy are computed from the predicted constituents and nine

samples are shown in Figures 5.2a and 5.2b. In the figures, it can be observed that the predictions capture the process of contraction, retraction, and to some extent convergence to a final RSAW. Though it is noted that in five of the samples the relative surface area starts to increase after 200-250 days again. In the figures, it can be seen that the prediction of the minimum RSAW is not accurate for the samples. The same holds for the strain energy with respect to the maximum. For the test set, the performance results for RSAW and the strain energy are shown in Table 5.2. The R^2-score of the RSAW and the strain energy distributions are negative, which means that the predictions are worse than reporting the average value. The aL_2-norms of 0.99 for RSAW and 534 for the strain energy also indicate that the overall error between the prediction and target is large. Therefore, it is concluded that for training on 15% of the data, the network has not learned the mapping of the time-step well enough to provide accurate predictions for RSAW and the strain energy over time. Even though the results on the batch of validation samples were good, it is clear that the errors for the prediction of the next time-step have a large effect when using iterative predictions.



(a) RSAW (b) Strain energy

Figure 5.2: Prediction for the RSAW and strain energy over 365 days using CNN, trained on 15% of the data, to predict constituents and mechanical values at each time step.

Performance	RSAW	E_{strain}
R^2	-0.91	-0.04
aL^2	0.99	534
aRelErr	14.2%	-

Table 5.2: Performance measures for prediction RSAW and E_{strain} 365 days ahead using iterative predictions with the trained CNN on 15% of the data.

5.2.4. Results 30% training data

To improve the performance, the network is trained on 30% of the training data, which results in a training set of approximately 457 000 samples. The performance on a 200 sample batch of the validation set is shown in Table 5.3. Since any differences in the performance could be due to validating on different samples, the results for the validation set are not compared to the results for training on 15% of the data. The R^2-scores for all constituents and mechanical values are above 0.98, which shows that the neural network has learned the distribution of these values quite accurately. The aRRMSE values below 0.1 indicate that the network can predict the next time step quite well. The aL_2-norms are dependent on the magnitude of the values and therefore it is expected that the values for M and N are higher. The reported performance on the validation batch indicates that the network is trained well. Again, it is more interesting to find the performance on the test set and to see whether the performance has improved.

For 240 samples in the test set the initial conditions, i.e. the constituents and mechanical values at $t = 0$, are passed to the neural network. The network is used iteratively to compute the predictions up and until day 365. Figure 5.3 shows the prediction and the target for the displacement u, the collagen concentration ρ and the strain ϵ at days 5, 50, 100, 250, and 365 for the best and worst sample in the test set. The predictions of RSAW and E_{strain} for nine random samples of the test set are shown in Figure 5.4. Table 5.4 reports the performance measures for the predictions of 240 samples in the test set. In Figure 5.3 it can be observed that the best predictions are accurate for almost all days,

Performance	M	N	c	ρ	u	v	ϵ
R^2	0.9913	0.9935	-	0.9942	0.9886	0.9822	0.9977
aRRMSE	0.0598	0.0782	-	0.0705	0.0381	0.0890	0.04588
aL^2	1364	4981	-	0.1067	0.1369	0.0125	0.0586

Table 5.3: Performance measures for the constituents and mechanical values on the validation set, using a trained CNN network on 30% of the data. Due to an error the values of c were not saved correctly.

although an increase for u can be observed on day 365. For the worst predictions, the prediction of u is slightly better, but the predicted values are still far off the targets. From this figure, it can not be concluded whether the performance has increased significantly. In Figure 5.4a it can be seen that for seven out of nine samples, the prediction shows contraction, retraction, and convergence to a final value. The convergence to the final value has improved with respect to training on 15% of the data, as in seven of the nine cases RSAW converges and does not increase at the end. This is promising as it shows that the network has learned the general long-term behavior better by training on more samples. Furthermore, it is observed that the network predictions on samples with slow contraction and retraction have improved as well. This shows that the network has captured more varying distributions. The fact that the final value is relatively accurate in five of the samples, whereas the minimum is not, indicates that the error does not necessarily accumulate due to the iterative predictions. The overall performance of RSAW distribution has increased significantly, reporting a R^2-score of 0.4014 instead of -0.91, a L_2-norm of 0.4535 instead of 0.99 and a relative error of 5.66% compared to 14.2%. These performance measures and the observations on the nine samples support the claim that the predictions of RSAW have improved. The R^2-score for the strain energy is still negative, though the average L_2-norm has decreased from 534 to 364, which shows that the predictions are closer to the targets.

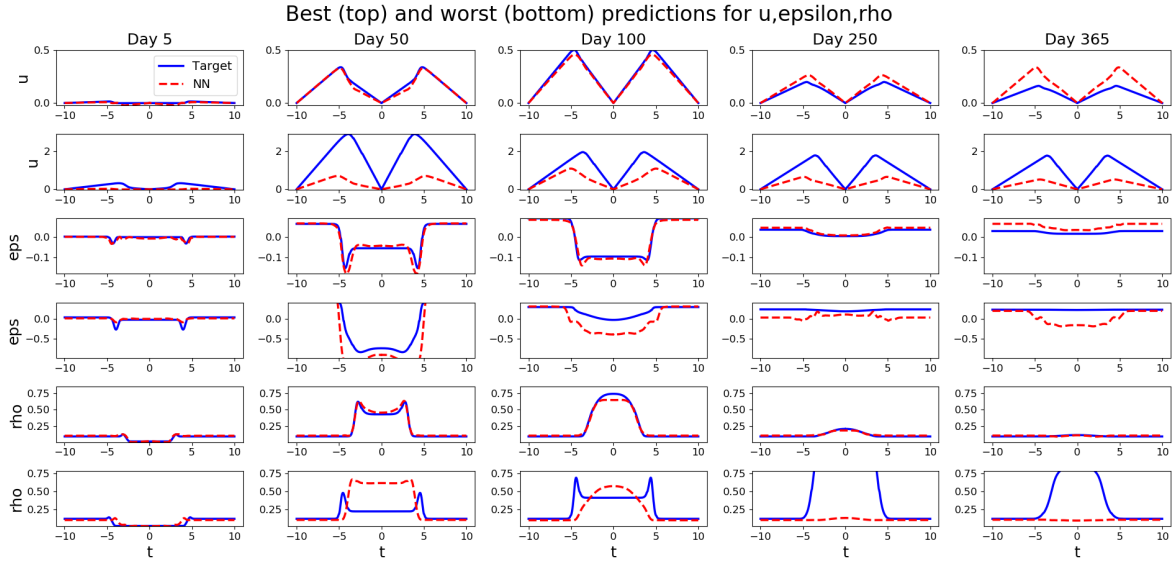
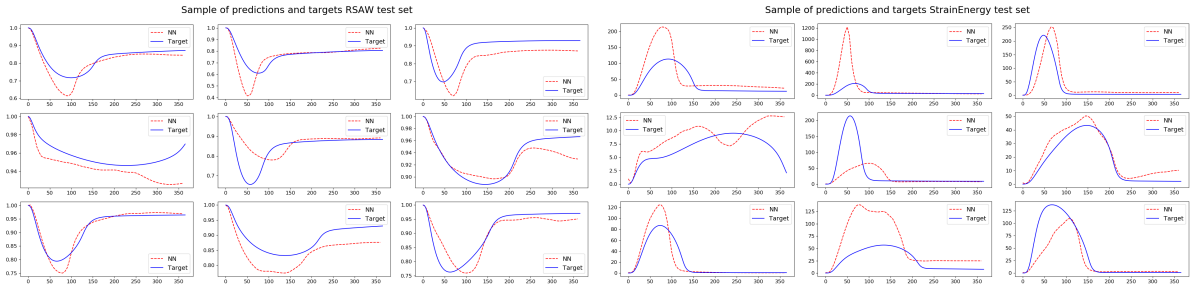


Figure 5.3: Best and worst predictions in terms of MSE for the displacement of the wound u, the concentration of collagen ρ and the strain ϵ at different time stamps for the test set.

Performance	RSAW	E_{strain}
R^2	0.4014	-0.2569
aL^2	0.4535	364.1
aRelErr	5.66%	-

Table 5.4: Performance measures for prediction RSAW and the E_{strain} 365 days ahead using iterative predictions with the trained CNN on 30% of the data.



(a) RSAW (b) Strain energy

Figure 5.4: Prediction for the RSAW and strain energy over 365 days using CNN to predict constituents and mechanical values at each time step.

To summarize, the main observations are that the performance on the validation set is good, that the network can capture the process of contraction, retraction, and convergence for 73 iterative predictions, and that the prediction of the RSAW and E_{strain} distributions are not accurate enough in terms of minimum and final values. From these observations, we reason that the network has learned the general mapping but not the effect of the 25 input parameters, which determine the specific distributions of RSAW and E_{strain}. Based on these observations and the fact that the performance has increased by adding more training samples, we believe that the hybrid method is promising. To improve the performance, the network should be trained on more data for a longer period of time and the hyperparameters should be tuned.

5.3. Conclusion

In this chapter, a first step is taken towards a hybrid approach, where a neural network learns to predict one time step ahead. In the hybrid approach, the trained network can be used iteratively to provide predictions for the next 365 days. A CNN is trained on 15% and 30% of the training data, respectively. It is found that the performance of the network on the test set is increased significantly by the increase in training data. It was observed that although the predictions of RSAW and the strain energy are not accurate yet, the iterative predictions capture long-term effects as contraction, retraction, and convergence to a final value. From these observations, we derive that the network has not learned to the correct effects of the 25 input parameters, though it has learned the general mapping of one time step. This supports the claim that the hybrid approach seems promising. More research needs to be done to improve training. It needs to be studied what performance can be reached when more training data or different architectures are used.

6

Clinical Case Studies

The main advantage of using a neural network as a surrogate for the morpho-elastic model is the option to provide fast simulations. In this chapter, two possible applications are discussed that demonstrate the usability of neural networks and support the claim that fast simulations are essential. First, the possibility for the neural network to (re)produce a parameter study is studied. There is large uncertainty in the values of the input parameters, e.g. how they differ per patient and their effect on the contraction. Being able to perform fast parameter studies can give more insight into the parameters and their behavior. In this chapter specifically the influence of age on the parameters and skin contraction is studied. For the sake of validation, the study with the neural network is compared to the same study performed with the morpho-elastic model. Secondly, we show a basic concept of an application that can be used to assist medical staff when treating patients with burn wounds. It applies the neural network to compute fast predictions on the healing process and the probabilities of contractures. The application uses Monte-Carlo simulation to account for the uncertainty in the input parameters for the patient.

6.1. Age study

Egberts et al. [22] performed an age study to find the influence of the patient's age on skin contraction. To this end parameters, dependent on age, were chosen for four different age groups. Performing these studies requires a significant number of simulations per group and is therefore computationally intensive. It is interesting to find if the neural network, trained on the one-dimensional dataset, can reproduce this study. The fast simulations of the neural network would allow for much more thorough parameter studies within less time. It is noted that the study by Egberts et al. [22] uses spatially varying parameters, whereas the neural network has been trained on spatially constant parameters. The reproducibility test with the neural network can also show whether using spatially constant instead of spatially varying parameters for the neural network changes the results. First, spatially constant parameters are considered for both the morpho-elastic model and the neural network to replicate the study as close as possible.

To maintain a clear distinction we call the study directly based on the simulations from the morpho-elastic model for each of the groups the 'simulation-based study' and the study with the neural network surrogate the 'NN-based study'. Since this research is based on a preliminary version of [22], parameter choices and results are slightly different.

6.1.1. Parameters

To investigate the influence of age on the parameters, four groups based on age were defined and the parameters were divided based on whether they depend on age or not. For the different groups, a basic Monte Carlo simulation is performed to assess the uncertainty in the input data. Based on these simulations, conclusions were drawn on the effect of age on the contraction of the wound. First, the chosen parameter values are reported after which the results are discussed.

In the study by Egberts et al. [22], the parameters are divided into five categories: parameters not dependent on patient or space, only dependent on patient, only dependent on space, dependent on both the patient and space and parameters dependent on other parameters. The parameters that are

Group	Age
1	0-10
2	11-40
3	41-70
4	71+

Table 6.1: Age groups

Parameter	μ	Dimension
$\bar{M}$	0	cells/cm^3
$\bar{c}$	0	g/cm^3
$\tilde{c}$	10^{-8}	g/cm^2
δ_M	0.06	/day
$r_F^{\max}$	2	-
η^I	2	-
η^{II}	0.5	-
$k_\rho^{\max}$	10	-
χ_F	$2 \cdot 10^{-3}$	cm^5/(cells day)

Table 6.2: Values for the parameters that not dependent on patient or space.

Parameter	μ	σ	Dimension
a_c^I	10^{-8}	$5 \cdot 10^{-10}$	g/cm^3
a_c^{II}	10^{-8}	10^{-10}	g/cm^3
a_c^{IV}	10^{-9}	10^{-10}	g/cm^3
ξ	0.044	10^{-4}	(N g)/(cells cm^2)
δ_c	$5 \cdot 10^{-4}$	10^{-5}	cm^6/(cells g day)
k_c	$3 \cdot 10^{-13}$	10^{-15}	g/(cells day)
ρ_t	1.09	0.1	g/cm^3

Table 6.3: Values for the parameters that not dependent on patient, but vary over the domain.

space-dependent are drawn using a lognormal distribution with a Karhunen-Loeve expansion, giving a smooth variation over the domain. The initial mean and standard deviation for the lognormal distribution are specified for each of the groups. It should also be noted that the parameter values should be within (or close to) the ranges of parameters the network has been trained on. All the parameter values can be found in Tables 6.2 - 6.5 and the description of the parameters and the references can be found in Appendix C.

Furthermore, three parameters are dependent on other parameters:

$$\mu_E = \frac{112}{\sqrt{\bar{\rho}}} \approx 350 \frac{\text{N}}{(\text{g cm})^{1/2}}, \tag{6.1}$$

$$q = \frac{\log(\delta_N) - \log(r_F[1 - \kappa_F \bar{N}])}{\log(\bar{N})}, \tag{6.2}$$

$$k_\rho = \delta_\rho \bar{\rho}^2. \tag{6.3}$$

For each simulation, parameters are drawn from the respective age group and a simulation for 365 days with a time step of one day was performed on a domain of 10 cm ($\Omega = [-10, 0]$) with an initial wound of 4 cm ($\Omega^w = [-4, 0]$).

Since the network is trained using spatially constant parameters, for the NN-based study the input parameters can not vary over the domain. Hence for these parameters the values are varied between simulations/patients only and are constant on the domain. The parameters are varied between simulations using the normal distribution with mean μ and standard deviation σ. Although all means are within the range of the parameters the network has been trained on, drawing from the normal distribution with the given standard deviations could result in some of the values being outside the training range when the mean is close to the range boundary.

Parameter	μ^1	μ^2	μ^3	μ^4	Dimension
$\bar{N}$	$1.8 \cdot 10^4$	$1. \cdot 10^4$	$1.1 \cdot 10^4$	10^4	cells/cm^3
$\bar{\rho}$	0.1200	0.1125	0.1050	0.0975	g/cm^3

Table 6.4: Parameters that vary along patients but are not varied over the domain.

Parameter	μ^1	μ^2	μ^3	μ^4	σ	Dimension
D_F	$1.1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$0.9 \cdot 10^{-6}$	$0.8 \cdot 10^{-6}$	10^{-8}	cm^5/(cells day)
r_F	0.914	0.898	0.868	0.832	0.0369	cm^{3q}/(cells^q day)
κ_F	$3 \cdot 10^{-7}$	$6 \cdot 10^{-7}$	$8 \cdot 10^{-7}$	$9 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	cm^3/cells
δ_N	0.0149	0.02	0.021	0.0215	10^{-4}	/day
k_F	$1.09 \cdot 10^7$	$1.08 \cdot 10^7$	$1.07 \cdot 10^7$	$1.06 \cdot 10^7$	$1.09 \cdot 10^7$	cm^3/(g day)
D_c	$3.1 \cdot 10^{-3}$	$2.88 \cdot 10^{-3}$	$2.66 \cdot 10^{-3}$	$2.44 \cdot 10^{-3}$	10^{-4}	cm^2/day
a_c^{III}	$2.3 \cdot 10^8$	$2.2 \cdot 10^8$	$2.1 \cdot 10^8$	$2 \cdot 10^8$	$5 \cdot 10^4$	cm^3/g
δ_ρ	$6.05 \cdot 10^{-6}$	$6 \cdot 10^{-6}$	$5.95 \cdot 10^{-6}$	$5.9 \cdot 10^{-6}$	$1 \cdot 10^{-8}$	cm^6/(cells g day)
μ	100	100	140	180	40	(N day)/cm^2
ζ	100	300	450	600	30	cm^6/(cells g day)

Table 6.5: Parameters that vary both along patients and along the domain.

6.1.2. Spatially constant parameters

First, we study the reproducibility using spatially constant parameters for both the simulation- and NN-based study to ensure that both receive the same type of inputs. Hence input parameters that do not depend on space are kept constant over all simulations and parameters that do depend on space are drawn from a normal distribution with mean μ and standard deviation σ for each sample and kept constant over space. For the simulation-based study, a dataset is generated containing 500 simulations per group. The computation of the 2000 simulations takes 13-14 hours using four cores in parallel. For the NN-based study likewise 500 input combinations are drawn per group and fed to the neural network. Here the trained neural network from Section 4.2.4 is used. Computing the 2000 predictions takes approximately 0.02 seconds using the neural network.

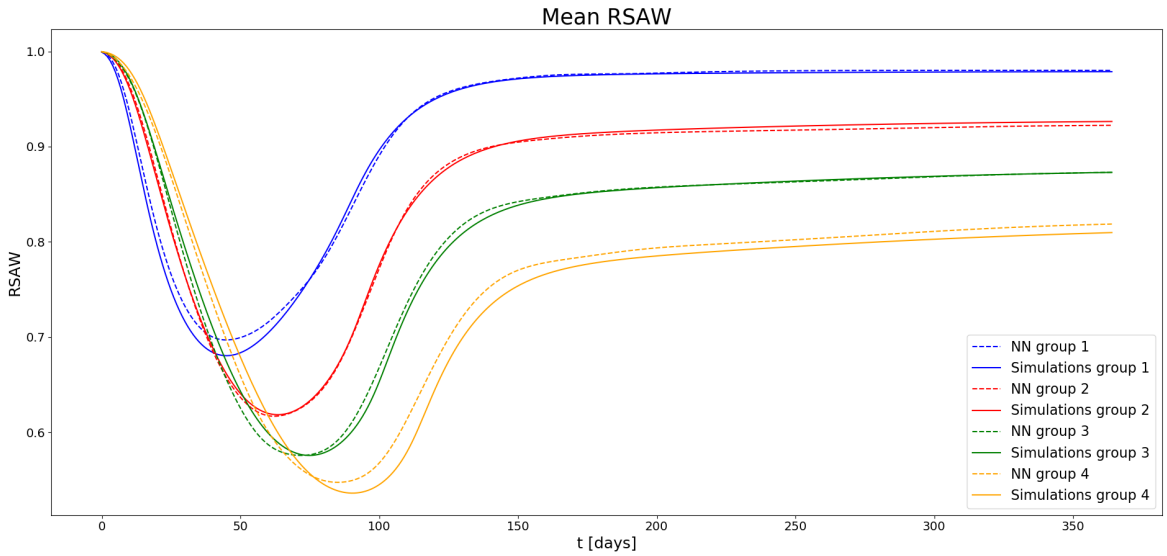


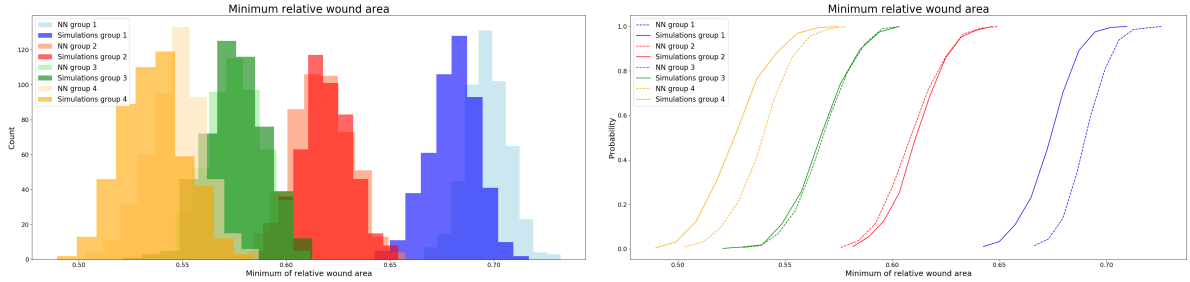
Figure 6.1: Mean RSAW for each of the four age groups. The results are shown for the simulation- and NN-based study (with spatially constant parameters from Tables 6.2- 6.5).

Group	MAE min RSAW	Rel err min RSAW [%]	MAE final RSAW	Rel err final RSAW [%]
1	0.0165	2.43%	0.0013	0.13%
2	0.0016	0.26%	0.0040	0.43%
3	0.00004	0.07%	0.00002	0.03%
4	0.0113	2.11%	0.0090	1.11%

Table 6.6: Performance on the minimum and final mean RSAW value for the four different groups in the age study with spatially constant parameters. (Using parameters from Tables 6.2- 6.5)

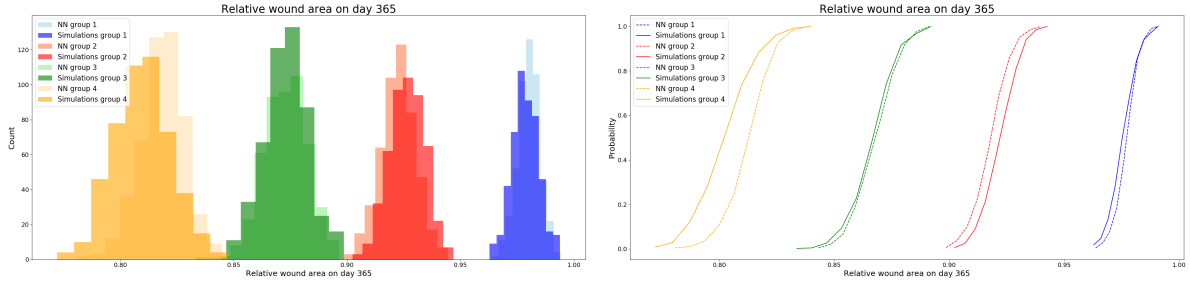
Figure 6.1 shows the mean RSAW for each group for both simulation-based and NN-based study. Table 6.6 shows the MAE and the relative error for the minimum and final values of the mean RSAW. Both Figure 6.1 and Table 6.6 show that results from the NN-based study are close to the results from the simulation-based study. For groups 1 and 4, the minimum RSAW it a little higher, and for groups 3 and 4 the minimum occurs slightly earlier. From the small MAE and relative errors it is evident that the results for groups 2 and 3 are very accurate. The relative error on the minimum values is below 2.5% for all groups and even below 0.3% for groups 2 and 3. For the final contraction, the relative errors are below 0.5% for group 1,2 and 3 and approximately 1% for group 4. Hence the neural network is able to reproduce the mean predictions of the age study accurately.

In terms of the age study itself, it can be observed that based on the predictions there is a distinction between the groups. As the age of the patient increases the contraction process takes more time, the maximum contraction is higher and there is less retraction resulting in a larger final contraction. For children, we find that the scar retracts almost completely leaving a small final contraction. For more detailed conclusions on the effects of age on skin contraction, we refer to the results in [22].



(a) Histogram (b) Estimate cumulative distribution

Figure 6.2: Results for the minimum RSAW for the four groups. Results from the simulations by the numerical model and the neural network are shown in the same figure. (Using parameters from Tables 6.2- 6.5)



(a) Histogram (b) Estimate cumulative distribution

Figure 6.3: Results for the final RSAW for the four groups. Results from the simulations by the numerical model and the neural network are shown in the same figure. (Using parameters from Tables 6.2- 6.5)

In Figures 6.2 and 6.3 histograms and estimates of the cumulative distributions are shown for the minimum RSAW and final RSAW respectively. These show the variations between simulations for both simulation- and NN-based studies. The estimated cumulative distributions can give an estimate of the probability that the maximum or final contraction is above a certain threshold and therefore the probability of a contracture. From the histograms and the cumulative distributions, it can be observed that the histograms of the simulation- and NN-based study show the same spread, and the estimated cumulative distributions have approximately the same slope. This shows that both simulation-based and NN-based studies show the same variability in the results. Furthermore, for groups 1 and 2 we find a clear distinction for the minimum RSAW with only minimal overlap. Between groups 2 and 3 there is some small overlap and between 3 and 4 there is a larger overlap, meaning the distinction between these groups in terms of the minimum RSAW is less evident. Conclusions on the significance of these distinctions between the groups can be found in [22]. For the final RSAW values, the groups are distinct for all four groups and the cumulative distributions are accurate.

From this study it can be concluded that the neural network is able to reproduce the parameter age study using spatially constant parameters with high accuracy. From both studies, the same conclusions can be drawn about the effect of age on the contraction. This shows that the neural network surrogate can be used for fast parameter studies. We do find that the results are more accurate for groups 2 and 3 than for groups 1 and 4. This could be caused by the fact that the mean parameter values for groups 1 and 4 are closer to the edge of the intervals the network was trained on.

6.1.3. Spatially varying parameters

Next, spatially varying parameters for the simulation-based study are considered. For each sample the input parameters are varied over space using a lognormal distribution with Karhunen-Loeve expansion. For each group, the mean and standard deviation from Tables 6.2 - 6.5 is used. For the neural network, the input parameters are also drawn from the same distribution to maintain the same variance. It must be noted that for the NN-based study the parameters can not be space-dependent and are varied along patients instead.

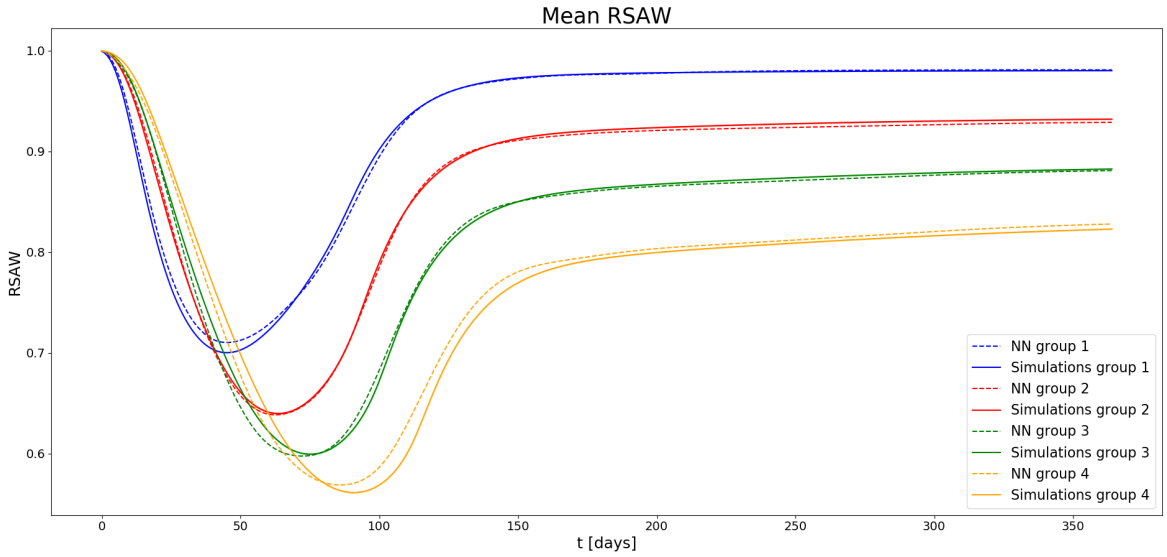


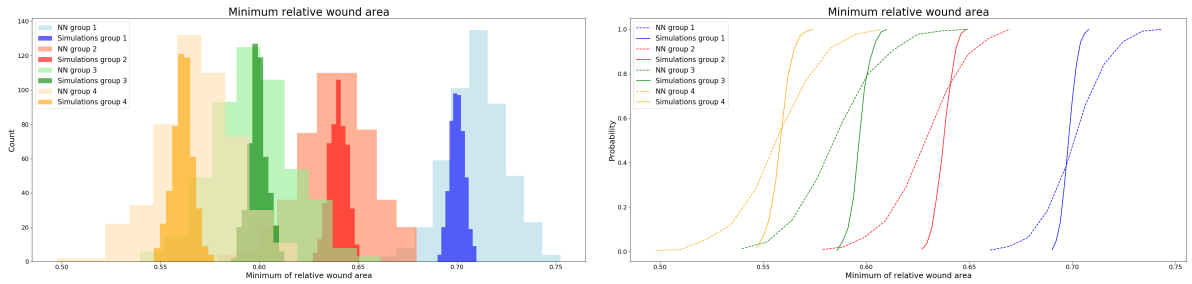
Figure 6.4: Mean RSAW for each of the four age groups. The results are shown for the simulations by the numerical model (with spatially varying parameters from Tables 6.2- 6.5) and the simulations by the neural network.

Group	MAE min RSAW	Rel err min RSAW [%]	MAE final RSAW	Rel err final RSAW [%]
1	0.0101	1.44	0.0007	0.07
2	0.0014	0.22	0.0032	0.34
3	0.0019	0.31	0.0016	0.18
4	0.0075	1.34	0.0050	0.61

Table 6.7: Performance on the minimum and final mean RSAW value for the four different groups in the age study with parameters with spatially varying parameters. (Using parameters from Tables 6.2- 6.5)

Figure 6.4 shows the mean RSAW for each group for both simulation-based and NN-based study. Table 6.7 shows the MAE and the relative error for the minimum and final values of the mean RSAW. Both Figure 6.4 and Table 6.7 show that results from the NN-based study are close to the results from the simulation-based study. For groups 1 and 4, the result is even 1% more accurate than for the spatially constant variation. Comparing to Figure 6.1, it can be observed that the minima of the simulation-based mean RSAW are slightly higher for group 1 and 4, and that the mean distributions for group 2 and 3 are very similar. This shows that using spatially varying parameters instead of spatially constant parameters does not significantly change the mean distribution.

In Figures 6.5 and 6.6 histograms and the estimated cumulative distributions are shown for the minimum RSAW and final RSAW, respectively. These show the variations between simulations for

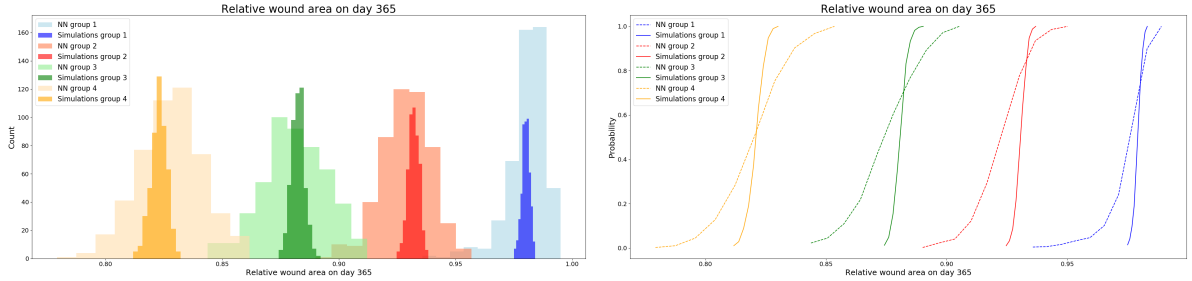


(a) Histogram

(b) Estimate cumulative distribution

Figure 6.5: Results for the minimum RSAW for the four groups. Results from the simulations by the numerical model and the neural network are shown in the same figure. (Using parameters from Tables 6.2- 6.5)

both simulation- and NN-based studies. It can be observed that the spread of the histograms for the simulation-based study is much smaller and the slope for the estimated cumulative distribution is much steeper. This shows that the variance for the simulation-based study is much smaller than for the NN-based study. The difference in variation can be explained due to the different variations in the parameters. In the simulation-based study, parameters are varied over space around the mean, such that each individual simulations has the same mean value over the domain. In the NN-based study, all parameters are constant over the domain and hence vary between simulations/samples, hence each individual simulation has a slightly different mean over the domain for each parameter. Using 0.5σ as the standard deviation for the NN-based study gives a similar variation as the simulation-based study. Hence we find that in case the Monte-Carlo predictions are only performed by varying the input parameters over space, a smaller variance should be chosen for the spatially constant neural network to obtain the same variance. However, if we vary the mean input parameters between patients/simulations, the spatial variance seems to be captured in the variance between simulation. For a more detailed study, one can compare the NN-based study to a simulation-based study with both patient and space variance.



(a) Histogram

(b) Estimate cumulative distribution

Figure 6.6: Results for the final RSAW for the four groups. Results from the simulations by the numerical model and the neural network are shown in the same figure. (Using parameters from Tables 6.2- 6.5)

6.1.4. Parameters outside training range

The previous values were chosen such that all parameters are drawn within the ranges the network has been trained on. However, it is interesting to find how well the network can give predictions when some of the values are outside the range. To that end another dataset of 500 simulations per group has been used to test this. In this set the values of $\bar{N}$, r_F, κ_F and a_c^{III} are changed such that for some of the groups they are drawn outside the training range. The values for group 2 are located on the edge of the training range, hence the variation causes some simulations to be within the training range and others to be outside the training range.

Parameter	μ^1	μ^2	μ^3	μ^4	σ	Training range
r_F	**1.222**	**0.924**	0.816	**0.611**	0.0369	$0.832 - 0.924$
κ_F	$0.8 \cdot 10^{-6}$	$\mathbf{1.5 \cdot 10^{-6}}$	$\mathbf{2 \cdot 10^{-6}}$	$\mathbf{3 \cdot 10^{-6}}$	$5 \cdot 10^{-7}$	$10^{-7} - 10^{-6}$
a_c^{III}	$2.01 \cdot 10^8$	$\mathbf{2 \cdot 10^8}$	$\mathbf{1.99 \cdot 10^8}$	$\mathbf{1.98 \cdot 10^8}$	$5 \cdot 10^4$	$(2 - 2.5) \times 10^8$
$\bar{N}$	$1.5 \cdot 10^4$	$\mathbf{10^4}$	$\mathbf{0.9 \cdot 10^4}$	$\mathbf{0.8 \cdot 10^4}$	0	$(1 - 2.5) \times 10^4$

Table 6.8: Parameters that have a different range with some values outside or on the edge the training range of the neural network. The bold values are outside or on the edge of the training range. The training ranges for the network are shown for comparison.

For the simulation-based study, 500 simulations were performed for each of the four age groups using the parameters from Table 6.8. The other parameters are equal to the previous section. For the NN-based study, likewise 500 inputs are drawn and fed to the trained neural network. Here the trained network described in Section 4.2.4 is used. Figure 6.7 shows the group means for both studies and

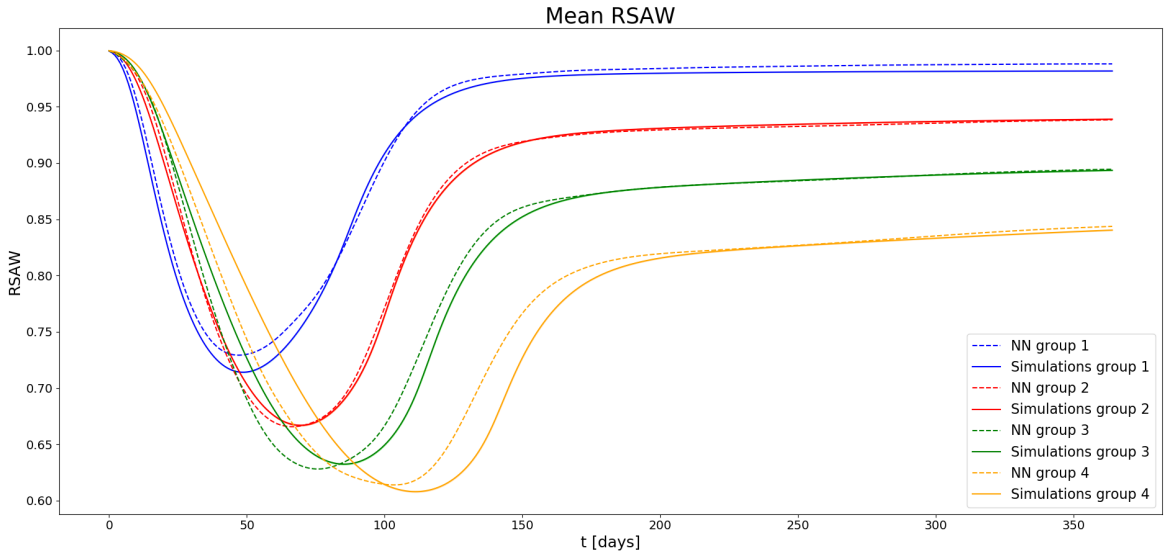


Figure 6.7: Mean RSAW for each of the four age groups with some parameters outside the training range. The results are shown for the simulations by the numerical model (with parameters from Tables 6.2, 6.3) and the simulations by the neural network.

Group	MAE min RSAW	Rel err min RSAW [%]	MAE final RSAW	Rel err final RSAW [%]
1	0.0151	2.12	0.0063	0.64
2	0.0013	0.19	0.0006	0.06
3	0.0044	0.69	0.0009	0.11
4	0.0060	0.98	0.0032	0.39

Table 6.9: Performance on the minimum and final RSAW values for the four different groups in the age study with parameters outside the ranges.

Table 6.8 shows the results on the minimum and final RSAW. It is observed that for group 1 the NN-based prediction of the minimum is 2% higher but on the correct day, which has similar performance to the spatially constant parameters within the training range. The prediction for group 2 is overall very accurate, showing little deviation from the simulation-based study. For groups 3 and 4, the predictions of the minimum RSAW values are accurate, however, they exhibit an earlier contraction and retraction mechanism, reaching the maximum and final contraction 10-15 days earlier. The prediction of the final contraction is accurate for all groups, with similar performance to the previous studies. From the accurate prediction of group 2, it can be concluded that for the chosen parameters drawing values close to the edge of the training range does not influence the result significantly. For the other groups, it is observed that the values outside the training range cause more deviation in the timing of contraction and

retraction. It is concluded that the results do become less accurate when some parameters are outside training range, however, the neural network can still provide reasonable generalizations. Though, it should be verified whether this holds for all input parameters and how far the parameters can be outside the range for reasonable predictions.

6.1.5. Conclusion

From the age study it can be concluded that the predictions by the neural network are accurate enough to study the influence of a patient's age on the surface area of the wound. In the case of the age study where the parameters means are within the training range, it was found that the resulting predictions of the minimum RSAW or maximum contraction have a relative error between 0.05-2.5%. The predictions of the final value have a relative error between 0.03-1.2%. It was also found that variance in the solutions due to the uncertainty in the input was similar for both simulation- and NN-based study. The neural network can accurately predict the estimated cumulative distribution for the maximum and final contraction. Furthermore, the effect of the spatial variance in the input parameters was studied. It can be concluded that the neural network can accurately predict the group mean RSAW without knowing the spatial variance. However, predicting the estimated cumulative distribution proved more difficult due to a larger variance in the NN-based predictions than the simulation-based predictions. The smaller variance in the simulation-based study can be explained since variation over the domain by Karhunen-Loeve expansion assumes the same mean for each patient, whereas varying per patient as in the NN-based study changes the mean. The effect of spatial variance should also be studied when combined with patient variance.

Lastly, we studied the effect of the parameter means that are outside or on the edge of the training range. For the four parameters that were varied, it was observed that values close to the edge of the training range did affect the prediction of the neural network negatively.

The simulation-based study needs 15 hours on four cores to compute 500 simulations per group. The network takes about 0.02 seconds to predict 500 simulations per group. Hence it can be concluded that the neural network surrogate can be useful for parameter studies, allowing to study many parameter combinations in a short amount of time.

6.2. Medical application

The knowledge about the contraction and the probability of a contracture can be valuable to medical staff when treating a patient. In case of a large probability that a contracture (that is a severe contraction) occurs the treatment needs to be different compared to when the likelihood is small. Furthermore, information on the strain energy in the wound during healing can give an indication of the amount of discomfort the patient experiences and can hence influence the treatment. Having access to this information can assist the medical staff in determining the best suitable treatment for each individual patient. In this section, an initial design of such an application is developed to conceptually demonstrate the potential of the neural network surrogate. In essence, the application reads the information from the patient and the wound and decides the distribution for the input parameters based on this information. The uncertainty in the input parameters is handled using Monte-Carlo simulation which requires many individual predictions for different sets of input values. The fast computation by the neural network allows us to compute these simulations in a very short amount of time. The results from the Monte-Carlo simulation are post-processed and visualized in the application. First, it is described how the distributions for the input parameters are decided, based on patient and wound information. Subsequently the post-processing of the results and the visualization are discussed.

6.2.1. Inserting input values

The user is asked to provide information about the patient and the wound, such as patient's name, age, wound size, and location. The age of the patient is used to determine mean values for the input parameters that depend on the age, similar to the age study in Section 6.1. The measured wound size is used directly as one of the input parameters. Furthermore, the location of the wound can be included as the elasticity of the skin can differ between locations on the body [47].

As was shown in the age study in Section 6.1, the network predictions are more accurate for input parameters that are within the ranges it has been trained on. Hence we use the ranges the network has been trained on as a basis for deciding the distribution of the parameters. For this demonstration,

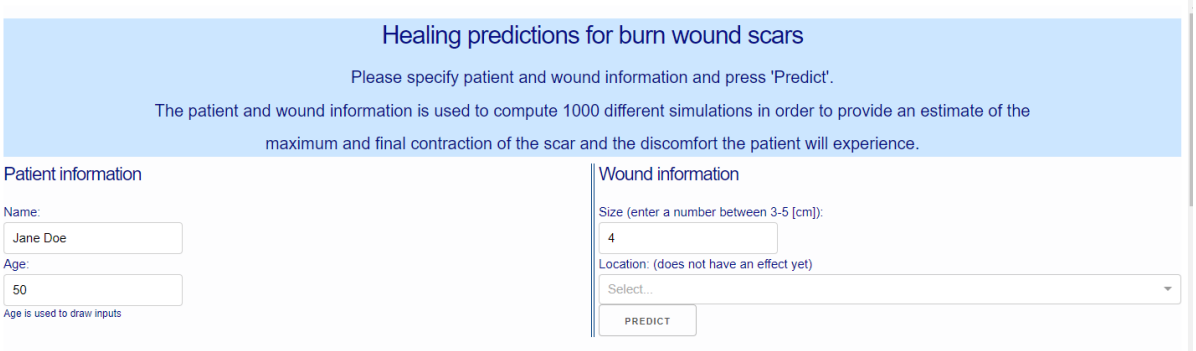


Figure 6.8: The input section of the application

only the effect of age on the parameter values is taken into account, using the information of the age groups from Section 6.1. When considering the age groups, is it reasonable to say that a 35-year old has parameter values closer to the group of 41-70-year-olds than a 15-year old, who might be closer to the group of 0-10-year-olds. This reasoning indicates a certain form of interpolation between the groups. Although it could be possible that certain parameters behave more step-like due to, for example puberty, we assume that the parameters can increase or decrease linearly over their respective ranges with increasing age. For each parameter a standard deviation is defined, for now these are assumed equal to Section 6.1. For the Monte-Carlo simulations, the input parameters are drawn from a normal distribution with the age-specific mean and the given standard deviation. For the age-independent parameters, the values are drawn from a normal distribution with a fixed mean and standard deviation. In total 1000 input combinations are drawn for each patient. The inputs are subsequently transformed with the fitted PCA-transform and scaled before being fed to the neural network.

6.2.2. Predictions & Visualization

For the predictions, two trained networks are used, one for the relative surface area of the wound and one for the strain energy. The transformed inputs are fed to both networks to obtain the predictions for RSAW and strain energy, resulting in 1000 predictions for each. Based on these predictions, the mean RSAW distribution, the 95%-confidence interval of the mean, and the standard deviation of the simulations from the mean are computed. The mean and its confidence interval are shown in blue in the application together with the interval $(\mu - \sigma,\ \mu + \sigma)$ in red. Furthermore, the estimated cumulative distribution can be computed from the simulations. Based on the cumulative distribution, an estimate can be given for the probability of a contracture, i.e. the probability of the final contraction passing a certain threshold. In the application we show the probabilities of a maximum contraction of more than 30% and the probability of a final contraction of more than 10%. The user can adapt the thresholds values which recomputes the probabilities. Figure 6.9 shows the visualization of the predictions for RSAW.

Moreover, the computed strain energy is shown to indicate the amount of discomfort. Analogously to RSAW, the mean and the 95%-confidence interval of the mean are shown together with the interval $(\mu - \sigma,\ \mu + \sigma)$. As it might be difficult to properly interpret the numerical values of the strain energy the graph is colored from white (little discomfort) to red (much discomfort) according to a fixed scheme. This scheme is chosen by the author for illustration purpose only. Figure 6.10 shows the visualization of the strain energy prediction. The run time of the application from pressing ‘Predict’ button until new visualizations appear is approximately 1-2 seconds. This proves that neural network surrogates can improve the applicability of the morphoelastic model for healthcare. Obtaining the same predictions with Monte-Carlo simulation by the morphoelastic model would have taken 6-7 hours. It is important to note that these characteristics and visualizations are chosen by the author for the means of illustration only and have not been tested or discussed with medical professionals.

The application is constructed using the Dash package in Python [3]. Using this package, the provided figure remains fully interactive, which increases the usability of the application. The application can be tested on the development server and is accessed via an internet browser. More information on the implementation can be found in Appendix B.

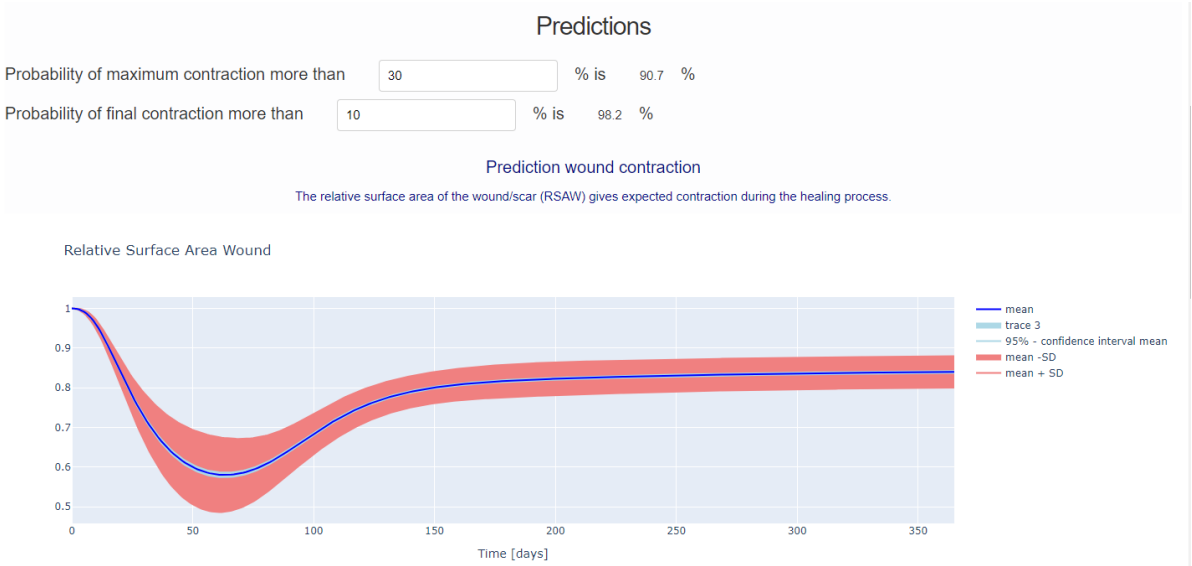


Figure 6.9: Visualization of the RSAW predictions in the application. The figure shows the mean and its 95-% confidence interval in blue. In red the interval $(\mu - \sigma,\ \mu + \sigma)$ is shown.

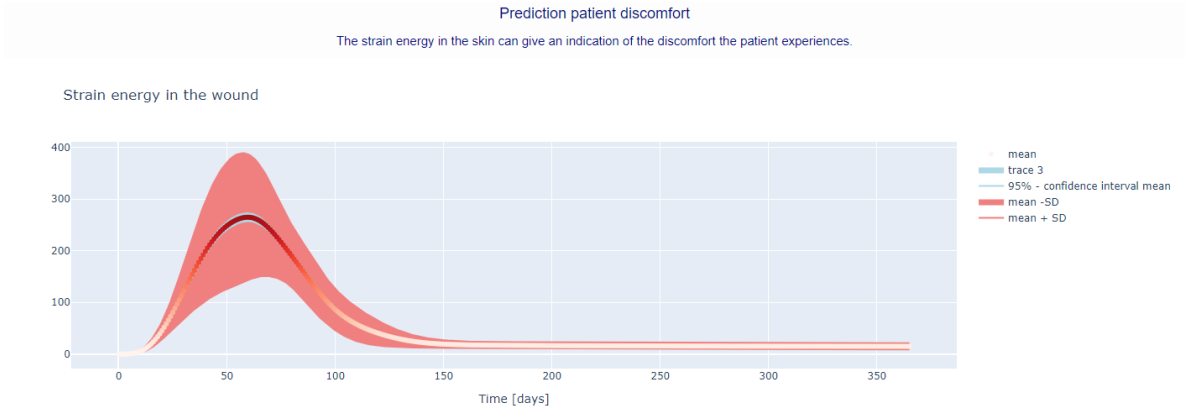


Figure 6.10: Visualization of the strain energy predictions. The figure shows the mean and its 95-% confidence interval in blue. The values of the mean are color coded along a fixed scale to give an indication of the amount of discomfort the patient will experience. In red the interval $(\mu - \sigma,\ \mu + \sigma)$ is shown.

7

Conclusion & Discussion

In this chapter, the conclusions of the study are presented and discussed. First, a concise summary of the research objective and the approach is given. Secondly, the conclusions on the direct surrogate model, the hybrid model and the clinical case studies are presented in Sections 7.1.1, 7.1.2 and 7.1.3 respectively. In Section 7.2, the methods and results of the study are discussed and recommendations for future research are given.

7.1. Conclusion

Burn injuries occur daily and can have severe physical and mental effects both in the short and long term, such as disabilities due to severe skin contraction. Even though the mortality rate has decreased over the years, the need for a higher quality of life after severe burns remains. Decreasing the probability of a severe contraction is essential for increasing the quality of life. Mathematical models have been designed to predict skin contraction over time. In these models, skin contraction is modeled by the use of a set of partial differential equations that involve several biochemical species, as well as a morphoelastic framework. The obtained systems are solved numerically using the finite element method. The simulations of the morphoelastic models are computationally expensive and are therefore less suited for applications where many simulations are necessary. We aim at a computationally cheap alternative modeling strategy that is based on neural networks. In the first approach, neural networks surrogates are trained to reproduce the nonlinear mapping between inputs and outputs of the morphoelastic models for skin contraction. The fast evaluation of the neural networks after training, achieves the required computationally cheap alternative. The surrogate approach is tested in varying scenarios for the prediction of the Relative Surface Area Wound (RSAW) and the strain energy. The second approach considers a hybrid model, where a CNN is trained to predict the computationally expensive computation of one time step in the morphoelastic model. The main advantages of the hybrid approach are larger model flexibility and access to all the involved biochemical constituents and mechanical parameters.

The main conclusion of the study is that a neural network surrogate is an effective and computationally cheap alternative that can reproduce the predictions of the morphoelastic model accurately. Furthermore, it is concluded that the use of the neural networks surrogate increases the applicability of the morphoelastic model for parameter studies and patient-based healthcare.

7.1.1. Surrogate model

In the surrogate approach, a neural network acts as a substitute for the complete morphoelastic model and is taught the entire nonlinear mapping between the inputs of the morphoelastic model and the desired outputs. The dataset for training and evaluation is generated from the one-dimensional morphoelastic model. It is shown that a two-layer MLP, trained to predict RSAW, can achieve high accuracy, reporting an average relative error of 0.39% and goodness of fit R^2 of 0.9932 on the test set. The important features of the skin contraction are the minimum RSAW, i.e. the maximum contraction, and the final RSAW, i.e. the final contraction. These characteristics were predicted accurately by the neural network, reporting a low MAE and a high R^2-score. The neural network needs only 0.008 seconds to

compute the predictions for 480 samples in the validation set. The computation of the original samples takes approximately 1.5 minutes per sample, which shows the significant acceleration neural networks can achieve. The trained neural network is also validated for combinations of input parameters that were unstable in the morphoelastic model. It was shown that the network can generalize well and that it can provide reasonable predictions for these samples as well.

Furthermore, it was found that the use of principal component analysis to reduce the dimension can be applied without any loss of information or reducing the accuracy of the prediction. When applying the PCA-transform, an explained-variance weighted loss function should be used to improve training. A second scenario for the prediction of RSAW is studied, where the displacement of the wound was predicted by the neural network and subsequently used to derive the RSAW instead of predicting RSAW directly. This scenario provided accurate solutions with an average relative error of 0.46% and goodness of fit R^2 of 0.9936. Combining the two different approaches in an ensemble, by averaging the two predictions, results in a slightly better prediction. It was shown that the difference between the two predictions in the ensemble has a weak to moderate correlation with the true error of the predictions, although we believe this correlation is not strong enough to provide an accurate estimate of the true error.

Besides the prediction of RSAW, another neural network is trained to predict the strain energy in the wound. A two-layer MLP reached a good performance with R^2-score of 0.981 and aRRMSE of 0.113. A second scenario was tested, where two neural networks were trained to predict the concentration of collagen ρ and the effective strain ϵ over time and space. From these predicted values the strain energy can be computed. Although the network performed slightly worse than the direct surrogate network, a good performance is achieved. The network performance for ρ and ϵ proved that a neural network can learn to predict the behavior of the constituents both over space and time accurately as well. The networks from the two scenarios were combined in a simple ensemble, by averaging the predictions. The network ensemble improved the performance slightly with respect to the individual solutions. It was also found that there is a moderate to strong correlation between the difference between the two individual predictions and the true error. Hence, using an ensemble in this scenario can improve the prediction and provide an estimate of the uncertainty of that prediction.

Furthermore, it was tested whether the RSAW and strain energy could be predicted simultaneously by one network. It was found that using a network with parameter sharing layers and private layers could reach similar performance as two separate networks for both outputs. This network performed better than the scenario where the RSAW and Estrain were combined and predicted by one fully shared network.

The standard surrogate approach was tested on a smaller dataset, constructed from the two-dimensional model as well. The performance of the trained networks for RSAW is slightly worse than for the one-dimensional model, though the predictions are still relatively accurate with R^2-score of 0.9577 and an average relative error of 0.74%. It was found that the network had significantly more difficulty learning the strain energy for two dimensions. As the number of samples in the dataset was limited, we believe that higher performance can be reached by increasing the number of samples. A visualization of wound shape evolution can be valuable for determining localized contractions and therefore a network is trained to predict the movement of the edge of the wound. The study shows that the network can learn to predict this movement accurately, although a larger number of training samples will further improve the performance.

From the studies, it can be concluded that neural networks can be used as fast and accurate surrogates for the morphoelastic models for skin contraction. Advantages of the surrogate approach are that the networks are easily trainable and give very fast predictions. A disadvantage of the surrogate method with respect to the morphoelastic model is that some of the flexibility and the physical interpretation of the model are lost. Another disadvantage is that the output format is fixed in the sense the network always predicts from day 0 to day 365. If these disadvantages with respect to the morphoelastic model do not pose a problem for the desired application, the surrogate neural network is an effective computationally cheap alternative for the simulations of skin contraction.

7.1.2. Hybrid model

A preliminary study for a hybrid approach is performed where a CNN is trained to approximate the non-linear mapping of an individual time step of the morphoelastic model. In the hybrid approach, the trained network can be applied iteratively to provide predictions for the next 365 days. Due to limited computational resources and limited time, the CNN is trained on 15% and 30% of the training data respectively. It is shown that the increase in training data significantly increases the performance of the model on the test set. From the study, it was observed that the trained network captures the long-term behavior of contraction, retraction, and convergence to a final contraction. The actual performance on the test set is not accurate though, with an R^2-score for RSAW of 0.4014. It is reasoned that it is likely the network has not yet learned the correct effects of the 25 input parameters, which are essential for the location and magnitude of the minimum and final contraction. To conclude, the proposed hybrid approach is far less accurate than the direct surrogate, though the results of this limited study show that the approach has potential. The networks need to be trained on all training data and different architectures need to be tested to provide well-based conclusions on the performance of the hybrid approach.

7.1.3. Clinical case studies

The two clinical case studies prove that the applicability of the morphoelastic model is increased significantly by using a neural network surrogate for fast prediction. A comparison of the parameter age study with the neural network to the study with the morphoelastic model proved that the same conclusions can be drawn on the influence of the patient's age on skin contraction. The results of both studies were similar and the computation time was decreased from approximately 15 hours to 0.02 seconds. This proves that the surrogate neural network approach allows us to explore the parameter space and their influences on the skin contraction much faster. The neural network is trained on spatially constant input parameters, while the morphoelastic model can handle spatially varying input parameters. It was shown that the lack of spatial variance for the neural network does not significantly influence the mean RSAW prediction, when compared to simulations by the morphoelastic model with spatially varying input parameters. However, the individual simulations of the morphoelastic model show less variance and a different estimated cumulative distribution, when input parameters are only varied over space and not between simulations.

In the second clinical case study, a concept application is developed, which demonstrates that the neural network surrogate can be used to give patient-based predictions based on the age of the patient. The fast predictions of the network allow for the uncertainty in input parameters to be handled by Monte-Carlo predictions. The developed application gives a clear example of how the neural network surrogate can be applied to provide clinicians with immediate access to simulations of skin contraction.

7.2. Discussion

In this section, the methods and results are discussed and directions for future research are provided.

Input parameter values

In order to provide input data for the training set for the neural network, the input parameters were chosen based on the preliminary results of the stability analysis and parameter sensitivity studies in [22, 23]. The obtained results on parameter sensitivity from the final study should be applied to improve the datasets. As mentioned, the values of these parameters are not easy to determine as they are not easy to measure or detect and can vary largely between individuals. The choice of the input parameters has a large effect on the prediction. Due to this uncertainty of the parameter ranges it is possible that certain predictions are not realistic. In this study, the input parameters are drawn uniformly from the defined ranges, which results in the desired variation between samples. However, the values of different parameters are likely correlated and the uniform distribution can result in unrealistic combinations of parameters and therefore unrealistic results. It should be investigated whether there are more representative methods of drawing the input values from the ranges.

Besides the ranges of the parameters, more research is needed on relations between different parameters and influences of factors like skin complexion, age, gender, and location on the body on the values of the input parameters. When these effects on the parameters are defined, the neural network surrogate can be used to study their effects on the skin contraction.

In the study for the two-dimensional model, the shape and size of the wound were considered to be fixed. It should be studied how to fit the shape and size of the wound in two dimensions to the network. Possible approaches are describing the initial wound as an image, which can be evaluated by a CNN to extract important features, or describing the wound shape by similarity scores to standard geometric objects, such as a circle or square.

Morphoelastic model

For this study, the simulations by the morphoelastic model have been taken as the truth. Naturally, these predictions are an approximation of the true wound behavior and have a certain accuracy as well. Hence, it is important to study the accuracy of the numerical model in more detail to give more detailed conclusions on the accuracy of the neural network for real-life applications. Furthermore, we encountered stability issues in the numerical model which need to be investigated further. The stability properties of the morphoelastic model studied in [23] should be applied to improve the dataset. The stability properties should also be investigated for the two-dimensional model.

For more accurate predictions the numerical models can be developed further to achieve more detail, more stability, and higher accuracy. The two-dimensional model used in this study is currently being developed and needs to be improved further. Moreover, a three-dimensional model can be developed to simulate the complete wound and provide more detailed predictions. It should be taken into account that the computational time of the morphoelastic models should be reasonable enough to enable generating a dataset for training a neural network within a specified time frame. Therefore, it is likely that other acceleration techniques need to be studied to achieve a reasonable run time. For example, utilizing the rotational symmetry can increase the efficiency of the three-dimensional model.

Meshing

The surrogate neural network approach relies heavily on the structure of the data and therefore the structure of the spatial and time discretization of the numerical model. For the two-dimensional model, this might become an issue when refined meshes are necessary around the wound edge and wound shapes show large variations. The application of CNN and pooling layers can provide a solution to handle the varying grids and extract the main features.

Physical interpretation

A disadvantage of using a neural network as a surrogate is the loss of physical interpretation of the results. Concentrations, for example, can turn negative in the predictions by a neural network. It should be studied if this can pose a problem and how it can be handled. It could be valuable to consider physics-informed or physics-guided neural networks [76] to improve the physical interpretation of the results.

Network ensemble

For the prediction of RSAW and strain energy, a simple ensemble of two neural networks was constructed by averaging the two individual predictions. The results were similar or slightly better than individual predictions. Therefore, the use of ensembles can be investigated further. In the study, only one type of ensemble is studied, which averaged the two predictions from different approaches. Ensembles can also be based on networks trained using the same approach, but different architectures or training algorithms [95]. Furthermore, it can be studied whether more networks in an ensemble can improve the prediction and whether there are better techniques to combine individual solutions. In the study, it was observed that the mean absolute difference between the two predictions gave a moderate correlation with the true error of the prediction. This correlation should be studied in more detail as it can provide a method for estimating the error of the prediction.

Hyperparameter optimization

The hyperparameters are the parameters that need to be defined before the training, including parameters for the network architecture, optimization algorithm and regularization techniques. The choice of hyperparameters is important for the final performance of the network. In this study, the hyperparameters have been chosen based on trial and error. More elaborate tuning of the hyperparameters could improve the performance results of the trained networks.

Furthermore, hyperparameter optimization techniques, such as Hyperopt [13], can be used to optimize the hyperparameters with respect to the validation loss.

Hybrid model

In this study, a first step is taken to develop a hybrid model where a neural network is used as a surrogate for the most time-consuming step of the model. Due to time and computational constraints, the performed study was limited and there is still a lot of research to be done on this topic. Firstly, the networks need to be trained on all the available data to improve the training, and validation should be performed on the complete validation set. Secondly, the hyperparameters should be tuned as different architectures and optimization algorithms can improve the performance of the networks. In terms of network architectures, the use of LSTM cells can be considered to ensure both long term and short term effects are captured.

The performance of this approach can be improved by following the approach in [33]. Here the change in state over one time step is approximated by the neural network instead of the full state at the next time step. Considering only the state change can improve the training as variations between different samples are smaller.

Furthermore, it can be considered to add physical relations, that the constituents should satisfy, to the loss function to improve the training. For example, one can add the error of an associated linearized system to the loss function. The addition of physical relations to the loss function can guide the training and reduce the risk of unrealistic predictions [48].

Recent studies on physics-informed learning have shown promising results, and it can be fruitful to investigate this approach for the application of a hybrid model. A physics-informed neural network can be used to quickly compute the solution to the derived system of equations [24, 76].

The training of the constituents and the mechanical values could be improved by scaling their distributions to a fixed range and saving the maximum/minimum of the distribution, using the approach taken in [95]. This method forces the network to learn the maximum and minimum values explicitly, which can improve the predictions.

Parameter study

In the parameter study, the effect of leaving out the spatial variance of the input parameters for the neural network was studied. To achieve a more fair comparison of the variance and the estimated cumulative distribution, the simulation-based study needs to be repeated with input parameters varying over space and between simulations. In this way, it can be observed whether spatial variance introduces additional variance that is not found when considering only patient variance. Furthermore, the effect of different parameters can be studied in more detail to find which parameters are most sensitive to the training range. The results of the sensitivity study by Egberts et al. [22] can be used to investigate whether the sensitive parameters in the numerical model are sensitive to the training range as well.

Medical application

In this study, the concept design of a medical application is developed, which provides a prediction of the contraction of the scar based on the age of the patient and the wound size. The effect of the age on the input parameters is interpolated linearly over age, which is likely a far too simplistic representation of this effect. Therefore, research is necessary to find the effect of age and location of the wound on the individual parameters to improve the choice of parameters for each patient. Furthermore, the uncertainty in the parameters should be studied to determine how much variation needs to be considered for a patient. The application should be extended such that it can use more patient and wound input to determine the distribution for the input parameters, such as degree of the burn, gender, and skin complexion.

Lastly, we note that the visualizations and displayed information have been chosen by the author for means of illustration and it should be discussed with medical professionals which information they need and how it should be visualized.

Bibliography

[1] World health organization. URL https://www.who.int/news-room/fact-sheets/detail/burns. (Accessed: 23-09-2020).

[2] Nederlandse brandwondenstichting. URL https://brandwondenstichting.nl/brandwonden/. (Accessed: 23-09-2020).

[3] Plotly Dash. URL https://plotly.com/dash/.

[4] H. Borchani, G. Varando, C. Bielzo, and P. Larranga. A survey on multi-output regresssion. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5, 2015.

[5] K. Alam and S.L.A. Jeffrey. Acellular fish skin grafts for management of split thickness donor sites and partial thickness burns: A case series. *Military Medicine*, 184:16–20, 2019.

[6] M. Alber et al. Integrating machine learning and multiscale modeling - perspectives, challenges, and opportunities in the biological, biomedical and behavioral sciences. *npj Digital Medicine*, 2 (115), 2019.

[7] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson. *The Molecular Biology of The Cell*. Garland Publishing, 2 edition, 1989.

[8] S. Amini and S. Mohaghegh. Application of machine learning and artificial intelligence in proxy modeling for fluid flow in porous media. *Fluids*, 4, 2019.

[9] S.A. Amrei, H. Ayatollahi, and S.H. Salehi. A smartphone application for burn self-care. *Journal for Burn Care & Research*, 41:384–389, 2020.

[10] B. Azzarone, C. Failly Crepin, L. Daya Grosjean, C. Chaponnier, and G. Gabbiani. Abnormal behavior of cultured fibroblasts from nodule and nonaffected aponeurosis of dupuytren's disease. *Journal of Cellular Physiology*, 117:353–361, 1983.

[11] V. Babovic, R. Canizares, H. René Jensen, and A. Klinting. Neural networks as routine for error updating of numerical models. *Journal of Hydraulic Engineering*, 127, 2001.

[12] A. Barion. An isogeometric analysis approach for morphoelastic models - application to skin contractures [master thesis]. Master's thesis, Delft University of Technology, 2020.

[13] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. Cox. Hyperopt: A python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8, 2015.

[14] D. Berrar. *Encyclopedia of Bioinformatics and Computational Biology*. Elsevier, 2019.

[15] D. Brinati, A. Campagner, D. Ferrari, M. Locatelli, G. Banfi, and F. Cabitza. Detection of covid-19 infection from routine blood exams with machine learning: A feasibility study. *J.Med Syst.*, 44, 2020.

[16] A. Buganza Tepole and E. Kull. Systems-based approach toward wound healing. *Pediatic Research*, 73(4):553–563, 2013.

[17] C. Calderón-Macías, M.K. Sen, and P.L. Stoffa. Artificial neural networks for parameter estimation in geophysics. *Geophysical prospecting*, 48:21–47, 2000.

[18] L.H.C. Chua and K.-P. Holz. Hybrid neural networks - finite element river flow model. *Journal of Hydraulic Engineering*, 135, 2005.

[19] A. Collette. *Python and HDF5*. O'Reilly, 2013.

[20] C.Yang, Y. Kim, S. Riyu, and G.X. Gu. Prediction of composite microstructure stress-strain curves using convolutional neural networks. *Materials and Designs*, 189, 2020.

[21] M. Despotovic, V. Nedic, D. Despotovic, and S. Cvetanovic. Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation. *Renewable and Sustainable Energy Reviews*, 56, 2016.

[22] G. Egberts, F. Vermolen, and P. van Zuijlen. A one-dimensional morphoelastic model for burn injuries: sensitivity analysis and a feasibility study. *arXiv:2010.12902*, 2020.

[23] G. Egberts, F. Vermolen, and P. van Zuijlen. A one-dimensional morphoelastic model for burn injuries: stability, numerical validation and biological interpretation. *arXiv:2010.12897*, 2020.

[24] M. Eichinger, A. Heinlein, and A. Klawonn. Stationary flow predictions using convolutional neural networks. *Technical Report Series Center for Data and Simulation Science*, 12 2019.

[25] F. Strutz et al. Tgf-β1 induces proliferation in human renal fibroblasts via induction of basic fibroblast growth factor (fgf-2). *Kidney International*, 59:579–592, 2001.

[26] M. Farage, K. Miller, and H. Maibach. *Degenerative Changes in Aging Skin*, pages 1–18. Springer, Berling, 2015. ISBN 978-3-642-27814-3.

[27] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[28] L.A. Garcia and A. Shigidi. Using neural networks for parameter estimation in ground water. *Journal of Hydrology*, 318:215–231, 2006.

[29] J. Gareth, D. Witten, T. Hastle, and R. Tibshirani. *An Introduction to Statistical Learning: with applications in R*. Springer, 7 edition, 2017.

[30] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 9, 2010.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Cambridge: MIT Press, 2017. ISBN 9780262035613.

[32] K. Gosh, Z. Pan, E. Guan, S. Ge, Y.Lio, T. Nakamura, Z. Ren, M. Rafailovich, and R. Clark. Cell adaptation to a physiologically relevant ecm mimic with different viscoelastic properties. *Biomaterials*, 28:671–679, 2007.

[33] R. Grzeszezuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proceedings of SIGGRAPH 98*, 1998.

[34] G. Gunin, N. Kornilova, V. Petrov, and O. Vasilyeva. Age changes in the number and proliferation of fibroblasts in the human skin. *Advances in Gerontology*, 1:299–303, 2011.

[35] X. Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, 2016.

[36] J. Han, A. Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *ArXiv*, 2018.

[37] Y.M.A. Hashash, S. Jung, and J. Ghaboussi. Numerical implementation of a neural network based material model in finite element analysis. *Numerical Methods in Engineering*, 59, 2004.

[38] A. Hatzfeld et al. Benefits of cryopreserved human amniotic membranes in association with conventional treatments in the management of full-thickness burns. *International Wound Journal*, 16: 1354–1364, 2019.

[39] J.M. Haugh. Deterministic model of dermal wound invasion incorporating receptor-mediated signal transduction and spatial gradient sensing. *Biophysical Journal*, 90:2297–2308, 2006.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv:1502.01852v1*, 2015.

[41] A. Heyneman, H. Hoeksema, D. Vandekerckhove, A. Pirayesh, and S. Monstrey. The role of silver sulphadiazine in the conservative treatment of partial thickness burn wounds: a systematic review. *Burns*, 41(7):1377–1386, 2016.

[42] and T.H. Le H.S. Tran and T.T. Nguyen. The degree of skin burns images recognition using convolutional neural networks. *Indian Journal of Science and Technology*, 9, 2016.

[43] M.G. Jeschke, M.E. van Baar, M.A. Choudhry, K.K. Chung, N.S. Gibran, and S. Logsetty. Burn injury. *Nat Rev Dis Primers*, 6, 2020.

[44] X. Jin, X. Zhang, K. Huang, and G. Geng. Stochastic conjugate gradient algorithm with variance reduction. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[45] I.T. Jolliffe and J. Cadima. Principal components analysis: a review and recent developments. *Philosophical Transactions A*, 374, 2016.

[46] A. Joshi, V. Shah, S. Ghosal, B. Pokuri, S. Sarkar, B. Ganapathysubramanian, and C. Hegde. Generative models for solving nonlinear partial differential equations. *NeurIPS*, 2019.

[47] A. Kalra, A. Lowe, and A. Al Jumali. An overview of factors affecting the skin's young's modulus. *Journal of Ageing Science*, 4, 2016.

[48] A. Karpatne, W. Wakins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv:1710.11431v2*, 2018.

[49] D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980v9*, 2019.

[50] D. Koppenol. Biomedical implications from mathematical models for the simulation of dermal wound healing [phd thesis]. Master's thesis, Delft University of Technology, 2017.

[51] D. Koppenol, F. Vermolen, and F. Niessen e.a. A mathematical model for the simulation of the formation of and the subsequent regression of hypertrophic scar tissue after dermal wounding. *Biomec Model Mechanobiol*, 16:15–32, 2017b.

[52] V.M. Krasnopolsky and F. Chevallier. Some neural network applications in environmental sciences. part ii: advancing computational efficiency of environmental numerical models. *Neural Networks*, 16:335–348, 2003.

[53] V.M. Krasnopolsky and M.S. Fox-Rabinovitz. Complex hybrid models combining determistic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, 19:122–134, 2006.

[54] N. Krueger and S. Luebberding. *Age-Related Changes in Skin Mechanical Properties in Textbook of Aging Skin*, pages 309–3167. Springer-Verlag Berin Heidelberg, 2017.

[55] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 1998.

[56] Z. Li and P. Maitz. Cell therapy for severe burn wound healing. *Burns & Trauma*, 6, 2018.

[57] C. Ma, J. Wang, and E. Weinan. Model reduction with memory and the machine learning of dynamical systems. *arXiv:1808.04258*, 2018.

[58] F. Meister, T. Passerini, V. Mihalef, A. Tuysuzoglu, A. Maier, and T. Mansi. Deep learning acceleration of total lagrangian explicit dynamics for soft tissue mechanics. *Computer Methods in Applied Mechanics and Engineering*, 358, 2020.

[59] S. Mohaghegh, C. Modavi, H. Hafez, M. Haajizadeh, M. Kenawy, and S. Guruswamy. Development of surrogate reservoir models (srm) for fast track analysis of complex reservoirs. *Society of Petroleum Engineers*, 2006.

[60] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty, and A.-B. M. Salem. Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, 3: 68–71, 2018.

[61] V. Moulin, G. Castilloux, F. Auger, D. Garrel, M. O'Connor-McCourt, and L. Germain. Modulated response to cytokines of human wound healing myofibroblasts compared to dermal fibroblasts. *Experimental Cell Research*, 238:1143–1170, 1998.

[62] V. Moulin, D. Mayrand, A. Laforce-Lavoie, S. Larochelle, and H. Genest. *Regenerative Medicine and Tissue Engineering - Cells and Biomaterials*. IntechOpen, 2011.

[63] K.E. Murphy, C.L. Hall, S.W. McCue, and D.L.S. McElwain. A two-compartment mechanochemical model of the roles of transforming growth factor β and tissue tension in dermal wound healing. *Journal of theoretical biology*, 272:145–159, 2011.

[64] K.E. Murphy, C.L. Hall, P. Maini, S. McCue, and D.L.S. MacElwain. A fibrocontractive mechanochemical model of dermal wound closure incorporating realistic growth factor kinetics. *Bulletin of Mathematical Biology*, 74:113–128, 2012.

[65] J. Navratil, A.J. King, J. Rios, G. Kollias, R. Torrado, and A. Codas. Accelerating physics-based simulations using neural network proxies: An application in oil reservoir modeling. *ArXiv*, 1906.01510v1, 2019.

[66] N.M. Nawi, M.R. Ransing, and R.S. Ransing. An improved learning algorithm based on the conjugate gradient method for back propagation neural networks. *Transactions on Engineering, Computing and Technology*, 14, 2006.

[67] D. Noble. The rise of computational biology. *Nature*, 3:460–463, 2002.

[68] L. Olsen, J.A. Sherratt, and P.K. Maini. A mechanochemical model for adult dermal wound contraction. *Journal of Biological Systems*, 3:1021–1031, 1995.

[69] World Health Organization. Burn prevention: Success stories and lessons learned. 2011.

[70] C. Overall, J. Wrana, and J. Sodek. Transcriptional and post-transcriptional regulation of 72-kad gelatinase/ type iv collagenase by transforming growth factor beta in human fibroblasts. *Journal of Biological Chemistry*, 266:14062–14071, 1991.

[71] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[72] M. Pawlaczyk, M. Lelonkiewicz, and M. Wieczorowski. Age-dependent biomechanical properties of the skin. *Advances in Dermatology and Allergology*, 5:302–306, 2013.

[73] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[74] D. Pond, A. McBride, L. Davids, B. Reddy, and G. Limbert. Microstructurally-based constitutive modelling of the skin - linking intrinsic ageing to microstructural parameters. *Journal of Theoretical Biology*, 444:108–123, 2018.

[75] M. Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *ArXiv*, 1804.07010v1, 2018.

[76] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *ArXiv*, abs/1711.10561, 2017.

[77] M.C. Reed. Why is mathematical biology so hard? *Notices of the AMS*, 51:338–342, 2004.

[78] A. Roberts et al. Transforming growth factor type β: Rapid induction of fibrosis and angiogenesis in vivo and stimulation of collagen formation in vitro. *Proceedings of the National Academy of Sciences*, 83:4167–4171, 1986.

[79] R.C. Rockne et al. The 2019 mathematical oncology roadmap. *Physical Biology*, 16, 2019.

[80] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv: 1706.05098v1*, 2017.

[81] B. Ráduly, A.G. Capodaglio, E. Lindblom, and K. Gernaey. Model reduction using neural networks applied to the modeling of integrated urban wastewater systems. *Proceedings 20th European Conference on Modelling and Simulation*, 2006.

[82] M.H. Sadd. *Elasticity: Theory, Applications and Numerics*. Elsevier Academic Press, 2009.

[83] T. Schaul, I. Antonoglou, and D. Sivler. Unit tests for stochastic optimization. *International Conference on Learning Representations*, 2014.

[84] J.A. Sheratt and J.C. Dallon. Theoretical models of wound healing: past successes and future challenges. *Comptes Rendus Biologies*, 325:557–564, 2002.

[85] J.A. Sherrat and J.D. Murray. Models of epidermal wound healing. *Proceedings of the Royal Society Biological Sciences*, 241, 1990.

[86] A. Sillman, D. Quang, B. Farboud, K. Fang, R. Nuccitelli, and R. Isseroff. Human dermal fibroblasts do not exhibit directional migration on collagen in direct-current electric fields of physiological strength. *Experimental Dermatology*, 12:396–402, 2003.

[87] M. Simpson, K. Lo, and Y. Sun. Age-related changes in pericellular hyaluronan organization leads to impaired dermal fibroblast to myofibroblast differentiation. *The American Journal of Pathology*, 175:1915–1928, 2009.

[88] C. Smolle, J.Cambiaso-Daniel, A.A. Forbes, P. Wurzer, G. Hundeshagen, L.K. Branski, F. Huss, and L. Kamolz. Recent trends in burn epidemiology worldwide: A systematic review. *Burns*, 43: 249–257, 2017.

[89] R. Snyderman. Personalized health care: From theory to practice. *Biotechnology Journal*, 7: 973–979, 2012.

[90] I. Spronk et al. Recovery of health-related quality of life after burn injuries: An individual participant data meta-analysis. *PLOS ONE*, 15, 2020.

[91] S.P. Timoshenko and J.N. Goodier. *Theory of elasticity*. McGraw-Hill, 1970.

[92] R.T. Tranquillo and J.D. Murray. Continuum model of fibroblast-drive wound contraction: Inflammation-mediation. *Journal of Theoretical Biology*, 158:135–172, 1992.

[93] J. Vande Berg, R. Rudolph, W. Poolman, and D. Disharoon. Comparitive growth dynamics and acting concentration between cultured human myofibroblasts from granulating wounds and dermal fibroblasts from normal skin. *Lab Invest*, 61:532–538, 1989.

[94] F.J. Vermolen. Chapter 1 - mathematical models of healing of burns. In *Innovations and Emerging Technologies in Wound Care*, pages 1 – 20. Academic Press, 2020. ISBN 978-0-12-815028-3.

[95] S. Wang, K. Fan, N. Luo, Y. Cao, F. Wu, C. Zhang, K.A. Heller, and L. You. Massive computational acceleration by using neural networks to emulate mechanism-based biological models. *Nature Communications*, 10, 2019.

[96] L. Wrobel, T. Fray, J. Molloy, J. Adams, M. Armitage, and J. Sparrow. Anex A: Table A.1. *Annals of the ICRP Publication 110*, 39:48–51, 2009.

[97] Y. Yu, H. Yao, and Y. Liu. Aircraft dynamics simulation using novel physics-based learning method. *Aerospace Science and Technology*, 87:254–264, 2019.

[98] R. Zou, W.-S. Lung, and H. Guo. Neural network embedded Monte Carlo approach for water quality modeling under input information uncertainty. *Journal of Computing in Civil Engineering*, 16, 2002.

A

Input parameter values

This appendix describes the values of the input parameters for the morphoelastic model in more detail. The parameter values are derived from literature by Egberts et al. [22]. Their findings are repeated here for completeness and extended to the choice of ranges for this study. It is noted that this study is based on a preliminary version of [22], which can cause values to differ slightly. For the two-dimensional model, a few parameter ranges had to be adapted to prevent numerical instabilities. For the relevant parameters, information on the effect of age is included to explain the choices in the age study in Section 6.1.

Symbol	Definition and derivation
$\bar{M}$	The equilibrium concentration of myofibroblasts in healthy skin. In Olsen et al. [68], the equilibrium concentrations are defined on the onset of the proliferation phase of wound healing. Therefore the initial myofibroblast concentration is set to zero everywhere: $\bar{M}$ = 0 cells/cm^3.
$\bar{c}$	The equilibrium concentration of signaling molecules in healthy skin. Taking into account the reaction term of the signaling molecules and the values of the other initial concentrations, $\bar{c}$ should be equal to zero [50]: $\bar{c}$ = 0 g/cm^3.
$\bar{N}$	The equilibrium concentration of fibroblasts in healthy skin is estimated by Olsen et al. [68] to be of $\mathcal{O}(10^4)$. Although other studies estimate a higher value of $\mathcal{O}(10^6)$, a equilibrium concentration of $\mathcal{O}(10^4)$ works better for the morpho-elastic model [22]. Therefore the range is defined as $\bar{N} = (1 - 2.5) \cdot 10^4$ cells/cm^3. Furthermore, Gunin et al. [34] found that number of fibroblasts in children skin (0-10 years) is nearly twice as high as adult skin (11+ years).
$\bar{\rho}$	The equilibrium collagen concentration in healthy skin is estimated by Olsen et al. [68]. Roughly 75% of the 15% of other substances than water and fat in 1 ml of human dermal tissue is collagen, which yields $\bar{\rho} = 0.1125$ g ml^{-1}. Egberts et al. [22] uses a slightly lower value of $\bar{\rho} \approx 0.1$ g/cm^3. Based on these results, the range is defined as $\bar{\rho} = 0.975 - 0.1200$ g/cm^3. Regarding the effect of age, Farage et al. [26] found that collagen content decreases at about 2% per year.
$\tilde{c}$	Initial concentration of signaling molecules in wounded skin. Due to the supply of growth factors in the inflammatory phase, the initial concentration of signaling molecules is unequal to zero. According to Olsen et al. [68], this value should not exceed 15-50 ng/ml, which leads to the range $\tilde{c} = (1 - 5) \cdot 10^{-8}$ g/cm^3
$\tilde{N}$	The initial concentration of fibroblasts in wounded skin is adopted from Koppenol [50], where it is estimated to be between 1000 and 2000 cells/cm^3, roughly 20% of the initial condition. Therefore the initial concentration is set to $\tilde{N} = 0.2\bar{N}$ cells/cm^3.
$\tilde{\rho}$	It is assumed no collagen is left in completely burned skin and therefore $\tilde{\rho} = 0$ g/cm^3.
χ_F	The chemotaxis coefficient is estimated in [64] to be $2 \cdot 10^{-3}$. This value is used as the lower bound for the range, which is defined by $\chi_F = (2 - 3) \cdot 10^{-3}$ cm^5/(cells day).

Symbol	Definition and derivation
κ_F	The division rate reduction value of the fibroblasts. Vande Berg et al. [93] approximates the carrying capacity of fibroblasts to be 10^{-6} cm^3/cells. In this study the value is estimated to be in the range $(10^{-7} - 10^{-6})$ cm^3/cells. Since the skin becomes thinner with increased age, it is assumed that crowding occurs faster in elderly and the value is increased with age [22].
η^I	The ratio of myofibroblasts to fibroblasts in the maximum secretion rate of the signaling molecule. Myofibroblasts produce roughly the twice the collagen that is synthesized by fibroblasts [61, 68]. This value is adopted: $\eta^I = 2$.
η^{II}	The ratio of myofibroblasts to fibroblasts in the secretion rate of MMPs which is estimated by Koppenol [50] to be 0.5. The value is adopted in this study.
δ_M	The apoptosis rate of myofibroblasts. A value of 0.06 /day corresponds to normal scars and 0.002 /day corresponds to hypertrophic scars [51]. This is combined with the study by Moulin et al. [62] which provides percentages, 8.85% for normal scars and 1.06% for hypertrophic scars. Combining the two studies leads to the range $\delta_M = 0.06 - 0.885$ /day.
δ_N	The apoptosis rate of fibroblasts. The average fibroblast doubling time (DT) ranges from 18-20h [7, 32] and the average lifespan varies between 40 and 70 population doublings (PD) [10, 62]. Egberts et al. [22] derived that this leads to the range $(0.0119 - 0.0231)$ /day, which is adopted. On average, doubling time decreases with age [87], which causes the apoptosis rate to increase with age.
δ_c	Proteolytic breakdown rate of the signaling molecules. Egberts et al. [22] derived the range $(3.54 - 6.93) \cdot 10^{-4}$ cm^6/(cells g day), which is adopted.
δ_ρ	Koppenol [50] estimates the value to be $6 \cdot 10^{-6}$ cm^6/(cells g day) and according to Farage et al. [26] the collagen turnover decreases with age. Therefore the value by Koppenol is used as an upper bound and the range is defined by $\delta_\rho = (4 - 6.075) \cdot 10^{-6}$ cm^6/(cells g day).
r_F	Proliferation rate of the fibroblasts. Egberts et al. [22] has derived that the proliferation rate is between 0.832 and 0.924 cm^{3q}/(cells^q day). This range is adopted for the study. Furthermore, the proliferation rate decreases with age [34].
$r_F^{\max}$	The maximum factor of cell division rate enhancement due to the signaling molecule is adopted from [25] and can range between 2 and 3.
a_c^I	Concentration of signaling molecule that causes half-maximum enhancement of the cell division rate. Olsen et al. [68] provided experimental evidence which indicates that half-maximal enhancement corresponds to concentrations about 10 ng per ml. We adopt the value 10^{-8} and vary this value with 10% to get the range $(0.9 - 1.1) \cdot 10^{-8}$ g/cm^3.
a_c^{II}	Signaling molecule concentration that causes half-maximum net secretion rate of the signaling molecules. We adopt the value from [68], where the value is related to the initial concentration of growth factors. Based on this result, the range is defined by $(0.98 - 1.02) \cdot 10^{-8}$ g/cm^3.
a_c^{III}	The value of the secretion rate for the generic MMP is estimated to be between 2 and $2.5) \cdot 10^8$ cm^3/g [70]. This range of values is adopted. It is assumed that this factor decreases with increasing age [22].
a_c^{IV}	Concentration of signaling molecules that causes the half-maximum enhancement of the secretion rate. We adopt the value 10^{-9} based on data from Roberts et al [78] and define the range by a deviation of $2 \cdot 10^{-10}$ g/cm^3.
k_F	The differentiation rate between fibroblasts and myofibroblasts. We adopt the range $5.4 \cdot 10^6 - 1.08 \cdot 10^7$ cm^3/(g day) computed by Egberts et al. [22]. Furthermore, the differentiation is decreased with age [87].
k_c	The maximum net secretion rate of the signaling molecule. From the stability analysis it followed that $k_c \leq \delta_c \bar{\rho} a_c^{II}$ [23]. To this end, the range is define as $(0.5 - 0.6) \delta_c \bar{\rho} a_c^{II}$ g/(cells day)
k_ρ	The secretion rate of collagen can give a stable reaction for the equilibrium when $k_\rho = \delta_\rho \bar{\rho}$ [22]. This result is adopted.
$k_\rho^{\max}$	The maximum factor of secretion rate enhancement due to the signaling molecules. Olsen et al. [68] found that the synergistic effects of growth factors may accelerate collagen biosynthesis up to tenfold. Hence we take $k_\rho^{\max} = 10$.
D_F	The diffusion rate of (myo)fibroblasts. Sillman et al. [86] studied the migratory rates of fibroblasts. In serum-containing medium the rate was $7.6167 \cdot 10^{-7}$ and in serum-free keratinocyte medium the rate was $1.86624 \cdot 10^{-6}$. This interval is adopted, i.e. $7.6167 \cdot 10^{-7} - 1.86624 \cdot 10^{-6}$ cm^5/(cells day).

Symbol	Definition and derivation
D_c	We define the range $(2.22 - 3.2) \cdot 10^{-3}$ cm^2/day for the diffusion rate of the signaling molecules based on the value derived in Haugh [39]. The assumption that the diffusion of signaling molecules decreases with age is adopted from Egberts et al. [22].
ρ_t	The total mass density of the dermal layer. Table A.1 in Wrobel et al. [96] shows that $\rho_t = 1.09$ g/cm^3 for human skin. The standard deviation of 0.1 g/cm^3 is adopted from Egberts et al. [22].
μ	The viscosity is estimated by Koppenol et al. [51] to be of order $O(10^2)$ and the stability analysis in Egberts et al. [23] shows that $\mu \geq \frac{\sqrt{\rho}E}{\pi}$. Hence values of μ between 10 and 1000 (N day)/cm^2 are considered safe.
R	A generic constant, which is estimated by Koppenol [50] to be 0.995 g/cm^3.
ξ	Generated stress per unit cell density. The value $\xi = 4.4 \cdot 10^{-2}$ (N g)/(cells cm^2) from [51] is used to define the range $(4.38 - 4.42) \cdot 10^{-2}$.
ζ	The morpho-elasticity factor is adopted from Koppenol et al. [51] to be between 0 and 900 cm^6/(cells g day). Krueger and Luebberding [54] showed that the skin's ability to recover after stretching decreases over lifetime and therefore the value is increased with age.
E	The parameter in the Young's modulus $Y = E\sqrt{\rho}$ is defined based on the equilibrium value of the collagen density. In Egberts et al. [22] the parameter is computed by $\frac{112}{\sqrt{\rho}}$. This value should be above 350 to achieve reasonable contractions. The value increases with age [72, 74]. The range is defined as $350 - \frac{300}{\sqrt{\rho}}$ N/((g cm)$^{\frac{1}{2}}$).
L	The initial length of the wound. The length of the wound is varied between three and five cm. This ensures the boundary conditions of zero displacement velocity remain valid.
q	The formula for q is derived from the fact that a stable chemical reaction is necessary when cell distributions and molecules are in equilibrium. We adopt the result from [22].

B

Implementation

Dataset

The datasets are constructed using the morphoelastic model, which is implemented in MATLAB. The implementation of the one- and two-dimensional model is done by Egberts et al. [23] and Barion [12] respectively and we do not provide further information on their implementation. The generation of the datasets is performed on a cluster of 27 nodes, where each node contains four or eight cores. For the one-dimensional model, each core of each node runs N_{core} simulations using a private seed for the random generator and a private file for saving the results. The private seed for the random generator is necessary to ensure all simulations have different input parameters. The cluster used MATLAB version 2015b. The datasets for the age study were run on a private laptop with Intel i7-8550U processor using four cores with MATLAB version 2018b.

Consecutively, the data is read into Python using the H5PY package which allows for fast access without the need to load the data into the memory [19]. The files are combined and split into a training and test set which are saved individually in hdf5 format. This is to ensure that test data is not used before the final test phase.

In order to efficiently load and access the training data, and to give easy access to the training and validation set, a *Dataset* class is constructed. This class opens the correct data file, training or test, and initializes the attributes. Additional functions are written to fit and save the transforms on the training set, transform the data and inverse transform the results. The class overwrites the *getitem()* function to return transformed inputs and targets, which allows using the *dataloader* class during training. The *dataloader* allows for the effective loading of the batches during training. Furthermore, the class is developed such that it is easy for the user to define one or multiple desired target variables. Different classes are developed for the surrogate datasets in Chapter 4 and for the hybrid dataset in Chapter 5.

The preprocessing of the data is done using the *sklearn* package [73]. Scaling is performed using the *StandardScaler* function and the principal components transform is fitted using the *PCA* function or the *IncrementalPCA* function. The *IncrementalPCA* function allows fitting the transform on batches of the data, preventing memory overload.

Neural networks

The neural network is implemented in Python 3.7 using the PyTorch library [71]. PyTorch is a high-level library, based on the Torch library, and is relatively easy to use. All training is performed on a private laptop with an Intel i7-8550U processor. The training of the neural network is performed on NVIDIA GeForce MX150 GPU to reduce training time using CUDA V10.2. To the end of organizing the training experiments, we make use of the Neptune platform and Neptune package for Python. All relevant values during training such as hyperparameters, training/validation loss, learning rate, and processor/GPU use, as well as performance measures, figures, and trained networks are saved to Neptune.

The neural network is implemented using a class that overwrites the standard *torch.nn.Module()* class with an initialization function to construct the network and a forward function to compute predictions for a given input. Algorithm 5 shows the general structure of the implementation performing cross-validation trials for a given neural network.

Algorithm 5: Pseudo-code for the cross-validation implementation.

```
Result: Performance network
Input: Data file, hyperparameters.
Initialize dataset;
for fold = 1:10 do
    Split dataset in training (90%) and validation set (10%);
    Fit scalers on inputs and targets of the training set;
    Initialize network, data loaders and optimizer;
    for i < N_epoch do
        for inputs,targets in train loader do
            Predictions = Network(inputs);
            Loss = Loss(predictions, targets);
            Backpropagate the loss;
            Adapt weights with optimizer;
        end
        for inputs,targets in valid loader do
            Predictions = Network(inputs);
            Loss = Loss(predictions, targets);
            if validation error < minimum validation error then
                Save network;
                Update minimum validation error;
            end
            else
                Check if early stop is reached;
            end
        end
        if Learning rate scheduler exists then
            Update learning rate;
        end
    end
    Post-process results validation set;
    Compute and save performance measures;
end
```

Hybrid model

The implementation of the training the neural network for the hybrid approach is similar to the training of the surrogate, using a different dataset and different network architecture. However, the implementation of the test set evaluation is different, due to the iterative predictions. Algorithm 6 shows the pseudo-code for the hybrid model, where *trainNN()* and *EvaluateNN()* are functions that train and evaluate the neural network. More detailed pseudo-code for the training and evaluation can be found in Algorithm 5. For the test set, 73 iterative predictions are used to compute RSAW and the strain energy for the next 365 days. In the algorithm p are the 25 input parameters of the morphoelastic model. The *Scale()* functions, apply the standardization scaler that is fitted on the training data.

Medical application

The medical application is implemented using the Dash package [3]. Using this package the lay-out of the application is designed using HTML and DCC components. The Dash package utilizes Plotly to maintain the full interactivity of the figures. The 'predict' button calls the main callback function which handles the Monte Carlo simulation, the pseudo-code of the main callback is shown in Algorithm 7. Furthermore changing the thresholds of the probabilities calls their own functions to update the probability based on the estimated cumulative distributions. Before the application is started, the networks and scalers for RSAW and strain energy are loaded.

Algorithm 6: Pseudo-code for training and evaluation of the hybrid model.

Result: Performance network
Input: Dataset, test set, hyperparameters.
Load (part of) the training and validation set;
Fit scalers on input parameters **p** and constituents inputs from training set;
Initialize network;
for *i* < N_{epoch} **do**
    TrainNN();
    EvaluateNN();
end
for *sample i* ***in*** *test set* **do**
    InputParam = Scale(p_i);
    InitCond = Scale([$M_i^0,\ N_i^0,\ c_i^0,\ \rho_i^0, u_i^0, v_i^0, \epsilon_i^0$]);
    Inputs = Concatenate(InputParam, InitCond);
    for *day* ***in*** *range(73)* **do**
        Preds$_i$ = [$M_i^{\text{day+1}},\ N_i^{\text{day+1}},\ c_i^{\text{day+1}},\ \rho_i^{\text{day+1}}, u_i^{\text{day+1}}, v_i^{\text{day+1}}, \epsilon_i^{\text{day+1}}$] = Network(Inputs);
        Inputs = Concatenate(InputParam,Preds);
        Inverse transform and save Preds;
        Compute and save RSAW$_i$(t) and $E_i^{\text{strain}}(t)$;
    end
end
Evaluate performance on the test set;

Algorithm 7: Main callback function application which performs the Monte-Carlo simulation for specified user input and updates the figures.

Input: N_{simul}, trained networks, maximum/final contraction thresholds
Read user input;
LoadInputs(N_{simul}, Age, WoundSize); {
    Define AgeFactor;
    $\mu_{\mathbf{p}}, \sigma_{\mathbf{p}}$ = input parameters(AgeFactor);
    inputs $\sim \mathcal{N}(\mu_{\mathbf{p}}, \sigma_{\mathbf{p}}, N_{\text{simul}})$ }
Transform(inputs);
RSAW,E_{strain} = Networks(inputs);
Inverse transform predictions;
Approximate cumulative distributions and compute probabilities;
Compute mean and standard deviations;
Update visualizations;

C

Hyperparameter tuning

The performance of a trained network is largely dependent on its hyperparameters, such as the architecture of the network, the optimization function, the learning rate, and the loss function. To find a combination of hyperparameters that gives good results for training, many combinations have been tested. As the dataset is prone to change due to improvements of the morphoelastic model and the parameter ranges, the hyperparameters have not yet been fully optimized, but merely tuned for good results. This appendix contains the conclusions for some of the trials that have been performed. It is noted that this is just an extract, many more trials have been performed during this study.

Activation functions
Three different activation functions, ReLU, $\tanh$ and sigmoid activation were tested. It was found that the sigmoid activation performs significantly worse than ReLU and $\tanh$. ReLU and $\tanh$ activation gave similar results in terms of final performance, but ReLU activation ensured faster convergence and thereby faster training. Based on this it was chosen to use ReLU activation units.

Optimization algorithms
Both optimization with SGD and with Adam was tested for different learning rates. From the trials, it was concluded that using the Adam optimization algorithm gives better results. The SGD algorithm also proved to be more sensitive to the learning rate, requiring more tuning than Adam. For Adam, multiple learning rates were tested between 0.001 and 0.01 with different decay factors. The Adam optimizer gave good results for most tested learning rates. Larger learning rates showed faster convergence, reaching a good performance within less epochs. The larger learning rate does have the effect that it can cause the algorithm to jump over the minimum. Adding a small learning rate decay ensured that the network can continue to learn for later epochs, as well. It is noted that using a smaller learning rate without learning rate decay can obtain similar performance when training for more epochs. For each new study, multiple learning rates between 0.001 and 0.01 and learning rate decays were tested.

Loss functions
In terms of loss functions, the MSE, the L_1-norm, and the Root Mean Squared Error (RMSE) were considered. In the study it was found that the L_1-norm performed worse than the other two loss functions. MSE and RMSE showed similar results. As the MSE is a built-in loss function in Pytorch and requires slightly less computations, it was decided to use the MSE loss function. For the MSE loss function it was studied whether weighted versions could improve the training. For example, by giving additional weight to the prediction of the minimum and final values, or by giving the loss of samples that are above a certain threshold a larger weight. These weighted versions did not significantly improve the result and did result in longer run time, and were therefore discarded. For learning the principal components, it was found that using an explained variance weighted loss function improved the result significantly.

Architectures

For the network architecture of the MLPs, different combinations for one-, two- and three-layer networks were considered. Layers consisted either of equal or varying number of neurons per layer. It was found that one-layer networks could reach reasonable performances, though two-layer networks performed significantly better in most cases. Using three-layer networks generically resulted in similar performance to the two-layer networks. For each network that predicted a different target, multiple architectures were tested.

For the hybrid approach, first a few MLP architectures were considered, though they performed significantly worse than the CNNs. For the network architecture of the CNNs, only a few architectures were considered. The considered architectures considered of one or two convolutional layers with varying number of output channels. Furthermore, architectures with one Maxpool layer after each convolutional layer and only one after the second convolution layer were considered. It was found that Maxpool layers after each convolutional layer decreased the dimensionality too much. It is noted that hyperparameter tuning for the hybrid approach was limited and will need to be done more extensively.

Regularization

For regularization purposes, it was found that early stopping was effective in preventing overfitting, since the gap between training and validation error remained small. Using dropout with $p = 0.1$ or $p = 0.2$ as regularization technique, lead to a slightly lower performance. Using L_1 or L_2-regularization also did not improve the generalization properties of the network with respect to early stopping. Since early stopping seemed to prevent overfitting, it was chosen only to use early stopping.