Universitetet i Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/Specialization: Engineering Structures and Materials – Mechanical Systems	Spring semester, 2021 Open / ~~Restricted access~~
Author: Mehrdad Saaedi	M.saaedi (Author's signature)
Faculty supervisor: Hirpa G. Lemu External advisor(s): Seyed Mohsen Mirkhalaf, Behdad Dashtbozorg	
Thesis title: **Predicting the Cancer Tumor Position in Liver Using Finite Element Analysis (FEA) and Artificial Intelligence (AI)**	
Credits (ECTS): 30	
Keywords: Soft tissue deformation FEM Machine Learning Artificial Neural Network	Number of Pages: xiv + 76 Stavanger, 15th June 2021

Summary

The computational power and advantages of the Finite Element Method (FEM) are noticeable. When dealing with high nonlinearity of the materials and geometrical complexity, FEM is a powerful solution, depending on the correct definition of the problem. The availability of this method has benefited many engineering areas. In the field of biomechanics and, more specifically, in Computer-Assisted Surgery, FEM is even more appreciated. This approach, however, comes at a high computational cost. Thus, a significant delay in the response impedes its implementation for real-time applications in clinical practices, even by using parallelization or utilizing Graphics Processing Unit (GPU). This is where an alternative approach is needed to accelerate FEM-based simulations to provide the desired outputs and minimizing the time lag, preventing using FEM during intra-operative applications.

A novel technique that may help to overcome the obstacles mentioned above and improve the response time is the field of Machine Learning (ML). In particular, the Artificial Neural Network (ANN), as a subset of ML, has demonstrated high potentials in computer vision and pattern recognition, whose implementation can be extended to replace a FEM model once it has been trained with sufficient inputs.

In this work, a FEM-ML framework is established to drastically increase the response time for predicting tumor and internal structures' locations in the human liver for surgical applications by using ANN. This technique takes advantage of the FEM results to train a model capable of capturing large deformations of liver tissue during the surgical intervention while reporting back the nodal locations of the components with high accuracy and efficiency. For doing so, a biomechanical model of the liver, accounting for the effect of the stiffness of blood vessels, is developed, and multiple simulations with random nodal loads on the surface of the liver are conducted in the commercial software Abaqus to produce the input required for the ANN. The ANN then predicts the nodes' coordinates resulting from the applied forces that can be used to reconstruct the deformed model of the organ.

Preface

This thesis is an endeavor to find the possible linkage between the areas of finite element method and machine learning in medical applications. This work has been conducted in collaboration with the University of Gothenburg and Netherlands Cancer Institute.

I would like to express my gratitude to Seyed Mohsen Mirkhalaf at the University of Gothenburg, Behdad Dashtbozorg at Netherlands Cancer Institute, and Hirpa G. Lemu at the University of Stavanger.

Further, a special thanks go to Ove Mikkelsen at the University of Stavanger for providing me with the license of Abaqus and helping with its remote usage.

Ultimately, despite the limitations, I am satisfied with the results of this work and the takeaway knowledge from it.

Table of Contents

List of Figures

List of Tables

List of Program Codes

Nomenclature

2D	=	Two-dimensional
3D	=	Three-dimensional
A_0	=	Two-dimensional
Adagrad	=	Aaptive Gradient Desscent
Adam	=	Adaptive Momentum Estimation
AI	=	Artificial Inteligence
ANN	=	Artificial Neural Network
b	=	Neuron's bias
BCs	=	Boundary Condintions
C	=	Right Cauchy-Green deformation tensor
CAS	=	Computer-Assisted Surgery
C_{ij}	=	Mooney-Rivlin model material parameter
CPU	=	Central processing unit
csv	=	comma-separated value
CT	=	Computed Tomography
det	=	determinent
D_i	=	Indication of incompressibility
DT	=	Decision Tree
E_{Euc}	=	Euclidean error
E^{r}	=	The output of training example
$E_{Relative}$	=	Relative error
ET	=	Extremely randomized trees
F	=	Deformation gradient tensor
$f(x)$	=	Similarity function
$f(z)$	=	Activation function
FE	=	Finite Element
FEA	=	Finite Element Analysis
FEM	=	Finite Element Method
GB	=	Gigabyte
GHz	=	Gigahertz
GPU	=	Graphics Processing Unit
I_i $(i = 1, 2, 3)$	=	Strain invariant
IVC	=	Inferior Vena Cava
J^{el}	=	Elastic volume ratio
K_0	=	Bulk modulus
L_0	=	Initial length
L_f	=	Final length
log	=	logarithm
LR	=	Linear Regression

MAE	=	Mean absolute error
MAE	=	Mean absolute error
MB	=	Megabyte
ME_{Euc}	=	Mean Euclidean Error
min	=	Minute
ML	=	Machine Learning
mm	=	Millimeter
MR	=	Magnetic resonance
ms	=	Millisecond
MSE	=	Mean squared error
N	=	Newton
n_{test}	=	Number of samples in the test set
p	=	Hydrostatic pressure
PC	=	Personal Computer
PDEs	=	Partial Differential Equations
R	=	Orthogonal finite rotation tensor
RAM	=	Random-access memory
ReLU	=	Rectified Linear Unit
RF	=	Random Forests
RMSE	=	Root mean squared error
RMSprop	=	Root Mean Square Propagation
SGD	=	Stochastic Gradient Descent
S^r	=	Input of training example
Stl	=	Stereolithography
SVR	=	Support Vector Regression
T	=	First Piola-Kirchhoff stress tensor
U	=	Right stretch tensor
V	=	Left stretch tensor
W	=	Strain energy function
w_i	=	Neural network's input weight
x_i	=	Neural network's input
$\hat{y}_i$	=	Predicted output
y_i	=	True value
z	=	Summed activation of the node
Γ	=	Learning rate
Γ_0	=	Initial learning rate
δt_i	=	Increment size
ϵ_{eng}	=	Engineering strain
$\lambda_i\ (i = 1, 2, 3)$	=	Principal stretch
μ	=	Shear modulus
ν	=	Poisson's ratio
$\sigma_i\ (i = 1, 2, 3)$	=	Principal Cauchy stress

Chapter 1

Introduction

1.1 Motivation and Background

The value of *Finite Element Method* (FEM) for solving complex engineering problems has been widely acknowledged. It has become a popular tool to numerically solve the governing partial differential equations rather than resorting to analytical methods. As for specific fields and applications like biomechanical engineering, due to the primarily non-linear nature of the materials, large deformations, and complex geometries, using FEM has been perceived as a popular approach [1].

In the field of medicine, one flow of the efforts is toward using less invasive methods with smaller and more precise incisions to reduce undesirable side effects of surgery such as the risk of bleeding and development of infection and subsequently reducing the patient's recovery duration. As of today, this is even more realistic by the availability of *Computer-Assisted Surgery* (CAS) tools that can help the improvement of surgical skills during preoperative training for the planning of the surgery or on the course of the surgery by predicting the internal structure of the organ for better navigation and guidance during the intervention. This is where a FEM-based biomechanical model can potentially be implemented to predict the intraoperative deformed shape of the organ. In this approach, the operating field is viewed by a laparoscopic camera, inserted through small abdominal incisions, and the operation is directed by watching the augmented view of the camera on a monitor [2], while the locations of internal components such as vessels and tumors are shown in real-time.

Nevertheless, a solution from a FEM simulation is a trade-off between the computational time and accuracy that both are of high importance during surgery that demands high precision and has a low tolerance for delay in the response. Therefore, embedding FEM in a real-time, accurate, and interactive system is very challenging. To reduce the computation time, several methods of using parallelism of the problem [3], using *Graphics Processing Unit* (GPU) [4], and dimensionality reduction technique [5] are among the most outstanding proposals in the literature. These techniques, however, to different extents lack enough accuracy for clinical applications and do not expedite the processes to a real-time level, achievable on general level hardware [6, 7].

Another different approach in addition to the aforementioned techniques that will be the focus of this work is the emerging field of *Machine Learning* (ML). Although the underlying basis of this approach dates back as early as mid-20th century, yet its dramatic progress did not begin until the past two decades with the arrival of high computing capacity [8, 9]. ML as a subcategory of *Artificial Intelligence* (AI), can take in a data-set consisting of several features (in the case of this thesis: applied loads, nodal displacements, material parameters, nodal coordinates, ...) and train a model based on the existing patterns between the features. This model is expected to anticipate specific values (in this study, the intended outputs are the nodal coordinates after deformation) for the unseen inputs fed into the model. This method, despite being expensive concerning the time it takes to train the model during the offline and preoperative stages, has given promising results concerning the requirements of real-time simulation and accuracy, and several authors have implemented this approach with success for different organs, and tissues such as brain [10], breast [11], and liver [2, 6, 7].

1.2 Aim and Scope

This thesis aims at utilizing and training an *Artificial Neural Network* (ANN) as a sub-field of ML to create a pipeline able to predict the shape deformation of the human liver by receiving the outputs of multiple FEM simulations from the commercial software Abaqus/-CAE. This can then be further developed to build an integrated, real-time, and interactive system assisting the surgeon in tumor localization during surgery.

To fulfill this endeavour, several questions will be raised and need to be properly addressed during this thesis:

1. How is the structure of a liver and how its different tissues behave?

   I What constitutive material models can properly describe the biomechanical behavior of these tissues?

   II What are the material parameters for these tissues/tumor?

2. What are the *Boundary Conditions* (BCs), the loads, and their magnitudes that the liver undergoes?

3. What type of element is a sound choice for this application?

4. What ANN architecture is suitable to use?

1.3 Thesis Layout

Concerning the aims mentioned above and the scope, this work contains six chapters. Chapter 2 tries to shed light on the liver structure and the theory of hyperelasticity as constitutive modeling for soft tissues and rubber-like materials. Moreover, this chapter aims at providing the necessary information about the finite element method and the basis behind machine learning with a focus on artificial neural networks. The reader can find information about the most recent findings and studies over the implementation of AI in the field of biomechanics in Chapter 3. In Chapter 4, the objective is to build a framework to perform the FEM simulations from the obtained information in Chapter 2 and Chapter 3. Furthermore, an initial setup of the required neural networks is executed, and the model is fed by the data gathered from the FEM. In Chapter 5, the results and findings of the preceding sections are assessed, and the ANN performance is evaluated. This thesis is concluded in Chapter 6, and the potentials for future works are also outlined.

1.4 Limitations

As mentioned before, the ultimate purpose of this work is to be able to use the end results to develop a computer-aided surgery platform capable of predicting the internal deformation of the liver from the organ's surface configuration as seen by a camera in real-time. However, several factors can potentially limit the objectives of this thesis as well as their extent.

1.4.1 Time

Due to the multidisciplinary nature of this thesis, tremendous efforts are required both in literature review and its technical aspect, which involves extensive computer programming and data acquisition. For this reason, a period of six months (the duration of the thesis) may not allow for real-world applicability of the outcome of this thesis, and inevitable compromises to contract the scope of this work might occur. For instance, reconstruction of the organ's model from the prediction of neural network's model is of high interest of this work yet out of the time scope.

1.4.2 Literature

This work relies heavily on the experimental results of previous researchers in the field of biomechanics. To the author's best knowledge, in certain areas such as biaxial test in compression of the liver tissue or mechanical properties of Glisson's capsule, due to the complexity of performing tests on biological tissues, there is a lack of solid and reliable experimental data. Hence, this work might have to settle for the existing material parameters extracted from the experimental data in uniaxial test results and disregard the effect of Glisson's capsule.

Chapter 2

Theory

As outlined in the previous chapter, this chapter intends to elaborate on the relevant theory and concepts inside liver anatomy, hyperelastic materials, a brief introduction of finite element analysis, and machine learning.

2.1 Liver Anatomy

In human, the liver is the largest internal organ that lies in the abdominal upper-right quadrant with a weight close to 1.5 kg [12]. The liver has specific functions ranging from detoxification of the metabolisms' products to bacterial removal from the blood [13]. This organ, partially covered by the lower ribs, is divided into right and left lobes by the *falciform ligament* that attaches the organ to the diaphragm and the abdomen's ventral wall [14]. The liver volume is also further split into eight segments by Couinaud classification, seven occupying the anterior and posterior volume of the liver, while the remaining one is located on the backside of the organ. In the following subsections, the main parts of the liver and their characteristics, important for a precise biomechanical model are briefly explained.

2.1.1 Parenchyma

The majority of the liver's building cells are hepatocytes, belonging to the parenchyma, which is the functional tissue constituting up to 80% of the total cells in the organ [15, 16]. Hepatocytes make up hepatic lobules, the liver's most minor functional units that have hexagonal forms. These lobules, in turn, are grouped to form the liver parenchyma.

2.1.2 Blood Vessels and Bile Ducts

The network of internal blood vessels (also referred to as the hepatic tree) in the liver is also of significant interest in biomechanics. This network includes the hepatic veins draining into the *inferior vena cava* (IVC) (the largest vessel in the body), hepatic artery connected to the abdominal aorta, and portal vein receiving blood for detoxification from the pancreas, spleen, gallbladder, and gastrointestinal tract [17]. Bile ducts are also another tubular network present in the liver that gathers the secreted (produced and discharged) bile in the liver and ultimately releases it into the small intestine [18]. The hepatic tree and the bile ducts are seen as the primary reason for the liver's heterogeneity [19]. In Figure 2.1, these different vessels can be seen.

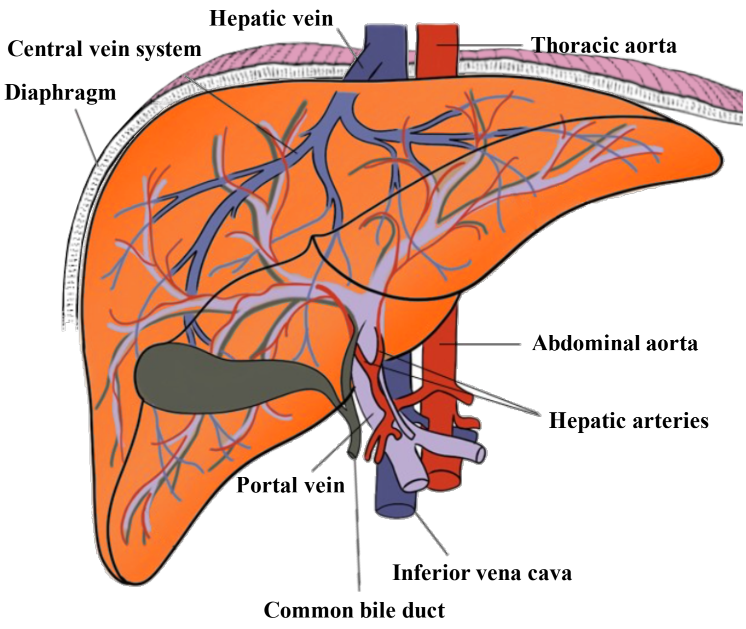


Figure 2.1: Liver's vascular system (Adapted from: Anatomy Note [20])

2.1.3 Glisson's Capsule

Another comparatively significant tissue in the liver that can affect the mechanical response of the entire organ is the Glisson capsule. This tissue is a collagen layer and, as indicated by Figure 2.2, surrounds the organ and ensheaths its vascular structure. The thickness of Glisson's capsule is species-dependent. In humans, however, it can vary from 70 to 100 μm [21, 22].

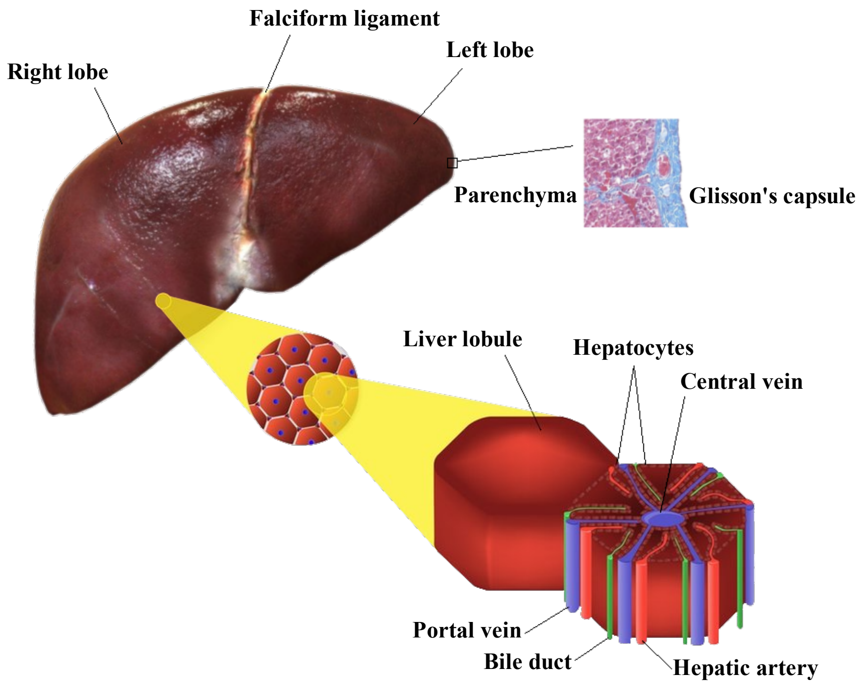


Figure 2.2: Liver's structure [23]

2.2 Nonlinear Constitutive Theories for Hyperelasticity

The liver has a relatively soft material whose mechanical behavior is characterized by the existence of different tissues that together act similar to a composite material. While various researches over the mechanical response of the liver have been conducted in high stretch rate regimes, minimally invasive interventions, especially laparoscopic surgeries, are more initiating lower rates of strain. Besides, during a clinical intervention, the organ may undergo several loading scenarios like shear, tension, compression, or a combination of these [24]. Therefore, to obtain reliable results from the FEA, the utilized constitutive model must account for the liver tissue's variety of stiffness and different loading conditions.

Due to the effect of the fluids (blood and bile) and porosity present in the parenchyma, the most realistic model for the liver is a visco-poro-hyperelastic material [6, 25]. A hyperelastic material for this tissue, though, can also be considered as a fair assumption for the slow rate of load application and minor *strain energy* dissipation [6, 26]. Ignoring the effect of viscoelasticity is tipically valid when static equilibrium is expected, and transient deformation is not taken into account [3].

2.2.1 Strain Energy Function

A *constitutive law* is an equation that relates physical quantities to a material's general characteristics. As opposed to Hook's law that decently describes the behavior of linear elastic materials, there is no universal formulation proficient enough to describe the non-linear behavior of all rubber-like materials [27]. Instead, many researchers have attempted to develop proper hyperelastic models that fit a more significant segment of the soft material's stress-strain curve with the assumption of deformations' reversibility. These models derive the relationship between the displacement and its corresponding stress from a strain energy function (W) [26].

Strain energy function (also known as stored-energy function) refers to the energy stored in the system as a result of deformation, which is released once the system goes back to its initial configuration. If the material homogeneity is assumed; meaning that the properties are the same at every point but may differ along different direction throughout the volume, then

$$F = \frac{\partial x}{\partial X} = \begin{vmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{vmatrix}, \tag{2.1}$$

where F is the deformation gradient tensor (second order) that provides the information to describe every current relative position of two particles in terms of their initial relative position, X is an arbitrary point in the reference configuration and x denotes the position of the same point in the current configuration. As shown in Figure 2.3 the deformation gradient F, which is the gradient of ϕ, transforms a material element dX (relative position between two particles in the reference configuration) into the corresponding spatial element vector dx which is the relative position of the particles in the current configuration [26, 28].

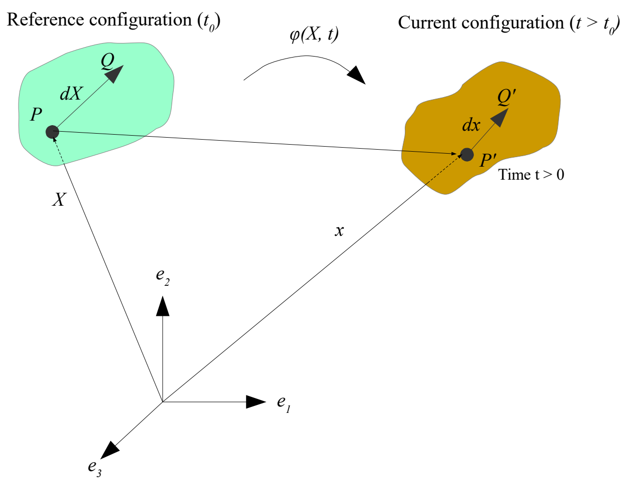

Reference configuration (t_0)
$\varphi(X, t)$
Current configuration ($t > t_0$)
Q
dX
P
Q'
dx
P'
Time t > 0
X
x
e_2
e_1
e_3

Figure 2.3: Transition from undeformed to deformed configuration (adapted from [29])

Based on polar decomposition, F can be decomposed into stretch and rotational component as

$$F = RU = VR, \tag{2.2}$$

where U is the right stretch tensor, V the left stretch tensor, and R is an orthogonal finite rotation tensor by which the rotation of eigenvectors U, N_i to the eigenvectors of V, $n_i : n_i = RN_i$ is represented [26]. The right Cauchy-Green deformation tensor C, which is a symmetric second order tensor, is a measure of the strain, the body experiences [30] and is derived from the deformation gradient tensor as

$$C = F^T F. \tag{2.3}$$

An invariant is a quantity that does not change under a specific mathematical/physical transformation or operation [31]. If the liver is assumed isotropic, i.e., the mechanical properties are not direction dependent but can differ from point to point, the strain energy function $W = W(F)$ is a function of three strain invariants of the deformation tensor I_1, I_2, and I_3 that are preferred instead of direct use of strain tensors [32] and are defined by

$$I_1 = trace(C) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \tag{2.4}$$

$$I_2 = \frac{1}{2}(I_1^2 - trace(C^2)) = (\lambda_1\lambda_2)^2 + (\lambda_1\lambda_3)^2 + (\lambda_2\lambda_3)^2, \tag{2.5}$$

and

$$I_3 = det(C) = (\lambda_1\lambda_2\lambda_3)^2, \tag{2.6}$$

where trace is an operator calculating the sum of the elements on the main diagonal as $\sum_{i=1}^{n} a_{ii}$, λ_1, λ_2, and λ_3 are principal stretches of deformation and the stretch tensors' eigenvalues [26, 27].

Since the liver is not highly confined within the abdominal cavity (otherwise, the degree of compressibility needed to be accounted for [33]), contains water (a nearly incompressible substance), and maintains its volume during deformation, it is a reasonable assumption to consider it highly incompressible. The mathematical interpretation of this is that the determinant of the Cauchy-Green deformation tensor (Eq. 2.6) for an incompressible material must be equal to 1.

2.2.2 First Piola-Kirchhoff Stress in Uniaxial Loading

From mechanics of material:

$$\epsilon_{eng} = \frac{\Delta L}{L_0} = \frac{L_f - L_0}{L_0}, \tag{2.7}$$

where ϵ_{eng} is the engineering strain, L_0 initial length and L_f is the final length. Since we know that $\lambda = \frac{L_f}{L_0}$, Eq. 2.7 can be rewritten to obtain the stretch ratio as

$$\lambda = \frac{\Delta L + L_0}{L_0}. \tag{2.8}$$

The 1st Piola-Kirchhoff stress tensor is defined as $T = \frac{F}{A_0}$, where A_0 is the initial area of the sample in tension or compression and F is the applied force. With the assumption of incompressibility, T, λ, and Cauchy stress are related by

$$T = \sigma\lambda^{-1}, \tag{2.9}$$

where for uniaxial deformation the stretch ratio is equal to the first principal stretch $\lambda = \lambda_1$ and Cauchy stress to the first principal stress $\sigma = \sigma_1$. Chui et al. [27] reported that in the uniaxial tensile test of a cylindrical liver sample, the sample had a $1/\lambda$ reduction of the cross-sectional area when the sample length showed an increase in height by a factor of λ. Therefore, By setting $\lambda = \lambda_1$, there will be $\lambda_2 = \lambda_3 = 1/\sqrt{\lambda_3}$, meaning that under uniaxial deformation the three invariants can be assessed as $I_1 = \lambda^2 + 2/\lambda$, $I_2 = 2\lambda + 1/\lambda^2$ and $I_3 = 1$. Hence, F is only a function of I_1 and I_2 [27]. The principal Cauchy stresses are also defined as [34]:

$$\sigma_i = \lambda_i \frac{\partial W}{\partial \lambda_i} - p, \quad (i = 1, 2, 3) \tag{2.10}$$

since $W = W(I_1, I_2)$, Eq. 2.10 is expanded by the chain rule for the first principal Cauchy stress as

$$\sigma_1 = \lambda_1 (\frac{\partial W}{\partial I_1}\frac{\partial I_1}{\partial \lambda_1} + \frac{\partial W}{\partial I_2}\frac{\partial I_2}{\partial \lambda_1}) - p, \tag{2.11}$$

and by having $\sigma_2 = \sigma_3 = 0$ (because of uniaxial tension/compression and no lateral force), hydrostatic pressure p can be calculated from

$$\sigma_2 = \lambda_1 (\frac{\partial W}{\partial I_1}\frac{\partial I_1}{\partial \lambda_2} + \frac{\partial W}{\partial I_2}\frac{\partial I_2}{\partial \lambda_2}) - p = 0. \tag{2.12}$$

Combining Eq. 2.11 and 2.12 leads to the removal of p from the general Eq. 2.10 and substitutes σ in Eq. 2.9 to derive the expression of the 1st Piola-Kirchhoff stress for uniaxial loading and the assumption of incompressible material as

$$T = \frac{2}{\lambda}\frac{\partial W}{\partial I_1}(\lambda^2 - \frac{1}{\lambda}) + \frac{2}{\lambda}\frac{\partial W}{\partial I_2}(\lambda - \frac{1}{\lambda^2}). \tag{2.13}$$

2.3 Hyperelastic Models

Several hyperelastic models have been proposed in the literature. These constitutive laws contain certain material parameters in their formulations, which are determined from the stress-strain curve derived from experimental test data like compression, elongation in uniaxial/biaxial, and shear tests. However, the data from the uniaxial tensile test of liver tissue are more readily available because of its relative simplicity in the procedure and measurement [24, 27, 35]. Considering that the vast theory behind all of the hyperelastic models and their several numbers is out of the scope of this work, this section only briefly explains *Ogden* [36] and *Mooney-Rivlin* [37] models and limits itself to the presentation of the strain energy functions (Table 2.1) of the other common constitutive laws used to describe the liver's mechanical behavior.

2.3.1 Categorization of Hyperelastic Models

Hyperelastic constitutive laws can first be categorized based on the form of their functions to polynomial, exponential and logarithmic, and combined exponential-polynomial. As illustrated in Table 2.1, they can also be classified based on their dependence on the Cauchy-Green tensor's strain invariants. In this classification, specific models such as Yeoh [38], Arruda-Boyce [39], and Neo-Hookean [40] only depend on I_1, and they are referred to as I_1-based models. For the characterization of the material represented by these models, only one type of test, e.g., uniaxial test, is required. Therefore, it is unrealistic to expect them to fully describe the behavior of a rubber-like material that may be prone to other deformation modes such as shear, biaxial extension, or compression [26]. However, under certain conditions, they might present an acceptable approximation. For instance, the Neo-Hookean model can provide a relatively decent fit with the experimental data for a low stress-strain regime. Mooney-Rivlin model is another invariant-based model that depends on both I_1 and I_2.

Another category of hyperelastic models (e.g., Ogden, logarithmic and exponential models) is directly based on the principal stretches. Invariants I_1 and I_2, therefore, are not seen in their functions.

Table 2.1: Main hyperelastic models used for the description of the liver's mechanical behavior and some prominent studies over their applications

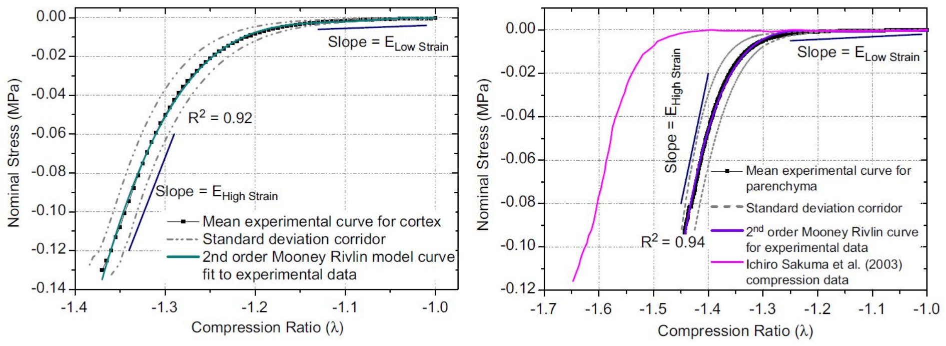
Model type	Model	Form	Function	Usage in literature for the liver's tissue
Invariant Based ($I_1 - I_2$)	Neo Hookean	Polynomial	$W = C_1(I_1 - 3)$	Chui et al. [27], Zaeimdar [41]
	Mooney-Rivlin (generalized)	Polynomial	$W = \sum_{i+j>0}^{N} C_{ij}[(I_1 - 3)^i (I_2 - 3)^j]$	Chui et al. [27], Hu and Desai [42], Fu et al. [43], Hostettler et al. [44], Umale et al. [35]
	Yeoh	Polynomial	$W = \sum_{k=1}^{N} C_k(I_1 - 3)^k$	Zaeimdar [41]
	Arruda-Boyce	Polynomial	$W = nk_B\theta[\frac{1}{2}(I_1 - 3) + \frac{1}{20N}(I_1^2 - 9) + \frac{1}{1050N^2}(I_1^3 - 27) + ...]$	Marchesseau et al. [12]
	Fung-Demiray	Exponential	$W = \frac{C_1}{2C_2}(e^{C_2(I_1-3)} - 1)$	Chui et al. [27], Roan and Vemaganti [45]
	Veronda-Westmann	Combined	$W = C_1(e^{C_3(I_1-3)} - 1) + C_2(I_2 - 3)$	Chui et al. [27], Yin et al. [46]
Stretch Based	Ogden	Polynomial	$W = \sum_{k=1}^{N} \frac{\mu_k}{\alpha_k}(\lambda_1^{\alpha k} + \lambda_2^{\alpha k} + \lambda_3^{\alpha k} - 3)$	Pellicer-Valero et al. [6], Martín-Guerrero et al. [47], Untaroiu and Lu [48], Chui et al. [27], Lorente et al. [49], Hu and Desai [42], Lister et al [50]
	Bogen	Polynomial	$W = \sum_{k=1}^{N} \frac{\mu_1}{\alpha_1}(\lambda_1^{\alpha 1} + \lambda_2^{\alpha 1} + \lambda_3^{\alpha 1} - 1)$	Chui et al. [27]
	Logarithmic	Logaritihmic	$W = -C_1 ln(1 - C_2(\lambda_1^{\alpha 1} + \lambda_2^{\alpha 1} + \lambda_3^{\alpha 1} - 3))$	
	Exponential	Exponential	$W = C_1(e^{C_2(\lambda_1^{\alpha}1 + \lambda_2^{\alpha}1 + \lambda_3^{\alpha}1)} -1)$	

2.3.2 Mooney-Rivlin Model

Mooney-Rivlin model is an instance of both invariant-based (I_1 and I_2) and polynomial form of the strain energy function that proposed by Mooney [37] and has been used in multiple studies to characterize the behavior of soft biological tissues such as liver's parenchyma, kidney [35, 51, 52] and brain [53]. The strain energy function for the Mooney-Rivlin model is given by

$$W = \sum_{i+j>0}^{N} C_{ij}(I_1 - 3)^i (I_2 - 3)^j, \tag{2.14}$$

where N is the order of the model and C_{ij} the material parameter. Figure 2.4 shows the implementation of the second-order of this model for liver and kidney, where the model's curve fits perfectly with the mean experimental curve.



(a) Renal cortex of kidney **(b)** Liver's parenchyma

Figure 2.4: Examples of stress-strain curves of second order Mooney-Rivlin model against compression test data, fitted for soft tissues [35].

2.3.3 Ogden Model

Ogden model, first derived in 1972 [54], has a stretch-based approach and, similar to the Mooney-Rivlin model, has a polynomial form. Due to the provision of a good fit with test data, this model has been widely used for modeling the liver's parenchyma and hepatic vessels [6, 27, 47–49, 55]. Ogden model is viewed as one of the most suitable constitutive laws for the description of incompressible, isotropic hyperelastic materials [56] and in its most comprehensive form is defined by

$$W = \sum_{k=1}^{N} \frac{\mu_k}{\alpha_k}(\lambda_1^{\alpha_k} + \lambda_2^{\alpha_k} + \lambda_3^{\alpha_k} - 3) + \sum_{i=1}^{N} \frac{1}{D_i}(J^{el} - 1)^{2i}, \tag{2.15}$$

where N is the model's order generally between 1-3, μ_k (shear modulus) and α_k (a dimensionless number) are material parameters, D_i is an indication of incompressibility that is defined by the bulk modulus K_0 as $D_1 = \frac{2}{K_0}$ and J^{el} is the elastic volume ratio [33].

In Figure 2.5, an example of a curved fit by the third order of this hyperelastic model can be seen. The curve plotted from the experimental data of the uniaxial test of liver tissue is also presented. As is apparent, decent proximity exists between the two curves.

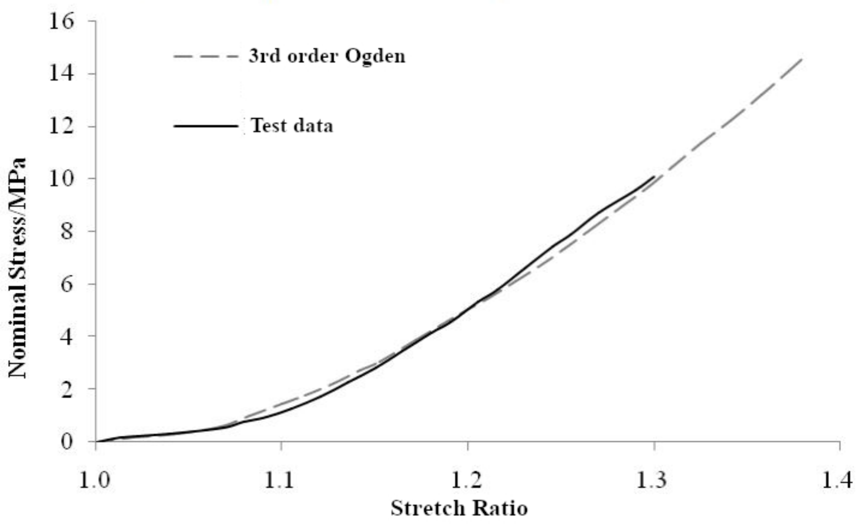


Figure 2.5: Stress-strain curve of Ogden third order and uniaxial tension test data of liver tissue [57]

2.4 Finite Element Analysis

Finite element is a numerical method for solving partial differential equations (PDEs) governing a physical problem that is difficult if solved analytically. As the naming also suggests, the foundation of this method is lying over using the discretization of the problem into smaller domains known as elements. It is essential to know that FEM only delivers an approximate solution to a problem whose accuracy depends on various parameters. The underlying bases of FEM, by which a solid mechanics problem is solved, is shown in Figure 2.6 and can be summarized as

- Compatibility
- Stress-strain relationship (constitutive law)
- Equilibrium

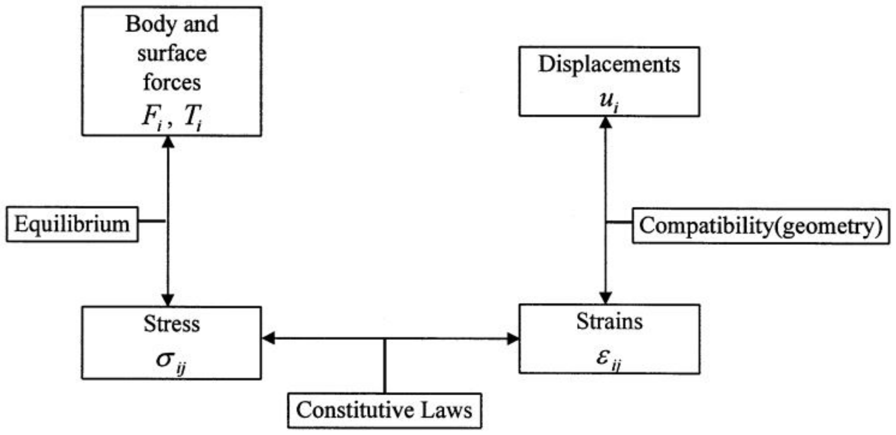


Figure 2.6: Underlying bases of FEA in solving solid mechanics problems [58]

Due to the vastness of the FEA theory, in this section, only the topics directly concerning the specific analyses performed in this thesis are reviewed. Unless otherwise stated, the theory presented in this section is studied from Abaqus User's Manual [33].

2.4.1 Nonlinear FEA

In contrast to linear FEA, a nonlinear analysis is more inclusive, and the stiffness and load matrices are not dependent on the displacement. This independence leads to the need for regular updates of the stiffness matrix throughout the analysis, and taking derivatives of the displacement does not necessarily result in finding the strains. In FEA, three sources that cause nonlinearity have been identified as

- Geometry
- Boundary
- Material.

Geometric Nonlinearity

This type of nonlinearity occurs when the magnitude of the displacement changes the response of the structure. The change in the initial shape, in turn, leads to the change in the stiffness of the structure under loading.

Boundary Nonlinearity

Boundary or contact nonlinearity takes place when interference or contact occurs between multiple parts. This interference causes a significant and sudden change in the structure's response, leading to the variation in the stiffness of the assembly.

Material Nonlinearity

This nonlinearity is the one expected to be the primary source in the analyses carried out for this thesis and occurs when the material itself is nonlinear in nature. In other words, the material's stress-strain curve does not show a linear relationship which is also observed in the behavior of biological soft tissues.

2.4.2 Elements

Two types of elements are expected to be used for the discretization of the liver tissue models. The first formulation, which makes up most of the elements in the parenchyma tissue, belongs to the continuum category of elements, which are solid with the hybrid formulation. The second element formulation is shell element, with the usage in modeling of parts with small thickness relative to the overall dimension of the tissues such as Glisson's capsule or the network of hepatic vessels.

Solid Element

Hexahedral and tetrahedral (Figure 2.7), are two general types of solid elements, suitable for discretizing three-dimensional parts. A hexahedron consists of six faces and eight corners, while 4 faces and 4 corners are the geometrical characteristics by which a tetrahedron is known. A geometry discretized with hexahedral elements generally results in a more structured mesh and thus higher solution accuracy. On the other hand, a mesh with tetrahedral elements can better fit geometries with complex features.

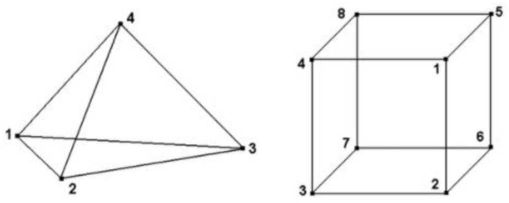


Figure 2.7: Tetrohedral and hexahedral elements

These elements can be further divided into elements with linear (first-order) and quadratic shape functions (second-order) categories, where in addition to the nodes located on the corner of the elements, each edge contains an extra node in the middle which is not shared by multiple edges. If large deformation is expected, the selection of a second-order element provides more accurate results at the expense of a higher computational cost.

Suppose the material is incompressible ($\nu = 0.5$) or nearly incompressible ($\nu > 0.475$) and uniform pressure (Figure 2.8) exists. In that case, since the volume of the element does not change, general formulations of solid elements cannot model the response of the material. Consequently, displacements of the nodes cannot be used for the calculation of pressure stress. For this situation, elements with hybrid formulation have been proposed that directly compute the pressure stress from an additional degree of freedom, and shear strains and stresses are computed from nodal displacements.

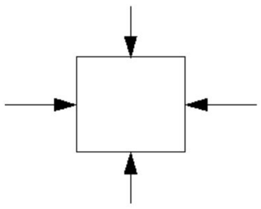


Figure 2.8: Uniform pressure

Shell Element

Using shell elements for members whose one dimension (e.g., thickness) is significantly smaller than the others can lower the computational burden. This is because these elements approximate a 3D space using a 2D theory, and dimensionality reduction can lead to a lower number of elements. In the Abaqus package, general-purpose 3D shell elements can be classified into triangular (S3) and quadrilateral (S4) elements. Furthermore, the linear or quadratic interpolation can also be used in S4 elements depending on the expected accuracy.

General-purpose shell elements ensure accurate and robust results under various loading scenarios and can be utilized in thin and thick parts, and shear locking is not a source of concern. These elements also allow for variation in thickness as a function of in-plane deformation.

2.4.3 Solution Methods

To find numerical approximation in finite element problems, the user can choose between two implicit and explicit methods. Although the simulations needed in this thesis are performed using an implicit solver, both methods are briefly introduced below due to the requirements of the problem.

Implicit

When dealing with nonlinear analyses, the time is discretized into finite increments known as time steps. In an implicit method, to find a solution, the displacements are formulated using velocities and accelerations in the current time step and the information from the previous step. One advantage of using this method is unconditional stability in most cases. Furthermore, large time increments can be used as long as large tolerance exists on the solution accuracy. This method, however, demands convergence checks at each step. This solution method is known to work best with various analyses such as linear and nonlinear static, heat transfer, and mass diffusion problems.

Explicit

As opposed to implicit, an explicit solver finds the solution for acceleration at the nth time step. When the values for accelerations are found, the velocity at n+1/2th step and displacement at n+1th can be calculated. Very small time increments are required for using an explicit solver, and the method is only conditionally stable. The explicit method is known for best handling high-speed dynamics, impact, buckling, and damage modeling problems [59].

2.5 Machine Learning

This section aims to give a short overview of the definition o machine learning and present the most common algorithms. A more in-depth exploration of the artificial neural network, which is the preferred technique to use in this thesis, will also be provided.

Machine Learning is a general concept enveloping all statistical algorithms developed to identify patterns and relationships between a given input instance, typically known as a dataset [60]. Machine Learning techniques are broadly categorized into three main classes of *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning*. In the following subsections these classes are briefly introduced.

2.5.1 Supervised Learning

Supervised learning algorithms are the most extensively used ML algorithms, working on the principle of finding a similarity function $f(x)$ that maps the given inputs to the known outputs. The model trained by this approach is then expected to predict the unseen inputs without any human assistance. As illustrated in Figure 2.9, the instances existing in the dataset used for training a model in a supervised learning algorithm are split in two by a user-defined percentage, e.g., 80 percent for training and the remainder for testing. Once the training, which usually is a time-consuming process, is done, the model's performance can be assessed by comparing the outputs it predicts for the descriptive features against the target features in the test dataset.

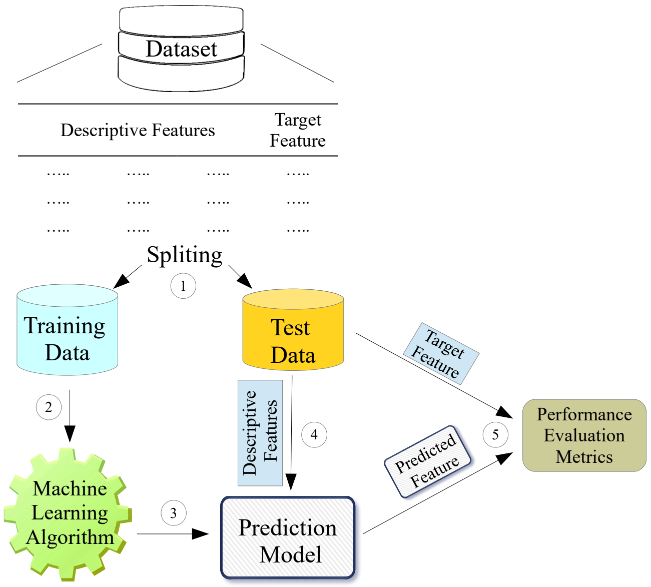
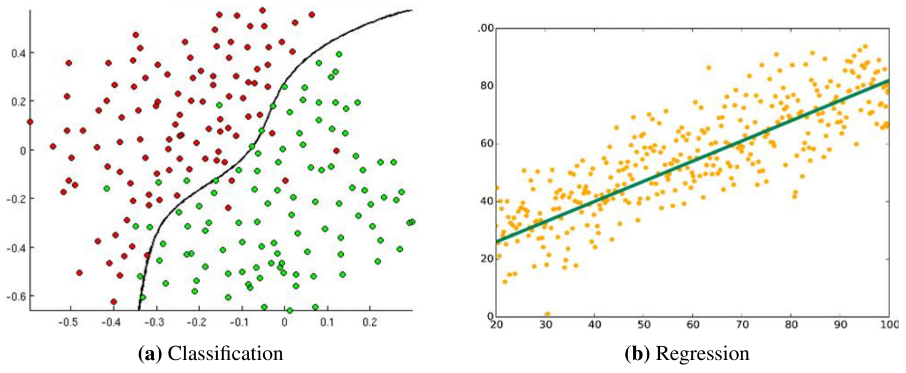


Figure 2.9: Learning and performance evaluation of a supervised ML model (adapted from [61])

Classification and Regression

Supervised learning is further divided into classification and regression tasks. In classification, the predicted outcome of the algorithm is qualitative (also referred to as categorical). The least complex type of these problems is the binary classification, where a label is either predicted as 0 or 1. In the example of predicting the presence of a dog in a picture, 1 can be associated with the presence, and 0 with absence [8]. In regression tasks, on the other hand, as also seen from the Figure 2.10 the quantitative output of regression algorithm is invariably a function, approximating the dataset's labels [8, 62].



(a) Classification **(b)** Regression

Figure 2.10: Classification vs. regression [63]

2.5.2 Unsupervised Learning

In unsupervised learning, the collected data is unlabeled, the output of the dataset fed to the learning algorithm is not known, and the algorithm does not receive any feedback. Nevertheless, it is still possible to learn the structure and pattern between the inputs [64]. In these algorithms fitting a linear regression model is impossible, and without a response variable y_i (an output), the algorithm analysis cannot be supervised [62]. An example of unsupervised learning is *cluster analysis* that aims at creating a new representation of the data by a distinct grouping of the input's variables. In the example of grouping personal photos uploaded into social media, for instance, without knowing which photo represents which person, the website algorithm can group the photos depicting the same person based on similar facial features [65].

2.5.3 Reinforcement Learning

Reinforcement learning algorithms are the third introductory class of machine learning. In these algorithms, unlike supervised learning, the need for paired input/output data is alleviated, while finding a balanced exploration-exploitation approach is emphasized [66]. In this area, the idea is to train an agent in interaction with a dynamic environment by receiving observations, sending actions, and receiving reward signals as a measure of success evaluation. As shown in Figure 2.11, reinforcement learning algorithm and policy are the terms referred to as the constituent components of the agent. By observing the environment, policy decides the actions, and the learning algorithm repeatedly updates the parameters of the policy in accordance to the sent actions and received observations and rewards in the hope of finding the most favorable policy that maximizes rewards of doing a task in the long run [67].

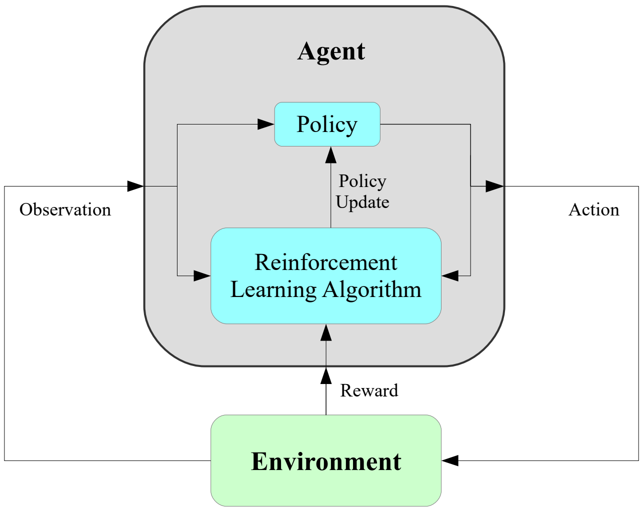


Figure 2.11: Reinforcement learning process adapted from [67]

2.6 Artificial Neural Networks

Artificial neural networks are a technique, falling under both supervised and unsupervised machine learning algorithms, that mimics natural biological intelligence. ANN comprises several to large numbers of interconnected elements, comparable to the biological neurons in the cerebral cortex of mammals (Figure 2.12), located on various layers that link the input information to the output. This subsection intends to present the key concepts and features required to understand the function and implementation of ANN as a subset of machine learning.

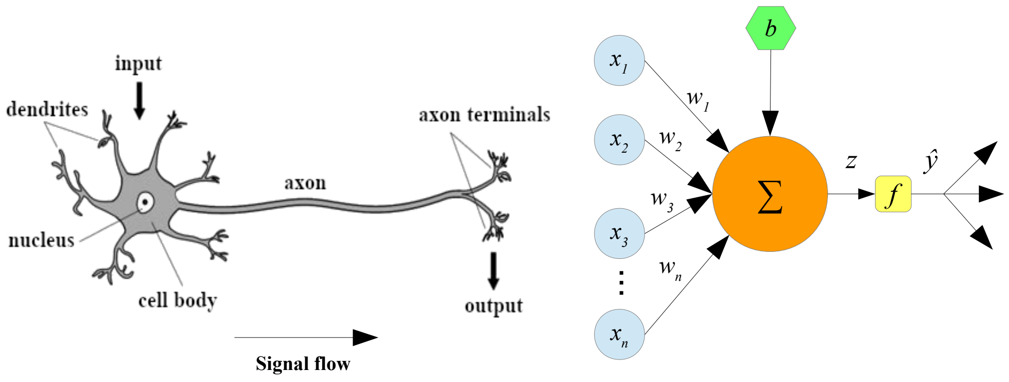


Figure 2.12: Biological and artificial neurons [68]

2.6.1 Perceptron Model

Being invented in the late 1950s by Frank Rosenblatt [69], perceptron[1] was initially hardware rather than an algorithm. Perceptron is the simplest form of a feedforward neural network containing one neuron whose synaptic *weights* w and *bias* b can be adjusted. In Figure 2.12, the perceptron takes inputs $(x_1, x_2, ..., x_n)$ and sums them up. However, to empower the perceptron with the ability to learn from the surrounding environment to correct the summed value and enable the predefined $\sum$ to take multiple shapes, weights $w_1, w_2, .., w_n$, must be multiplied to each input [8, 70]. Furthermore, assigning a particular bias to the neuron in a situation where input equals zero ensures that the neuron only receives nonzero quantities. Thus, the product of $x_i w_i$ must first overcome the value of the bias to affect the output before the activation function f. Therefore, a simple summation is defined as

$$z = \sum_{i=1}^{n} x_i w_i + b, \tag{2.16}$$

where n is the number of inputs and z is known as summed activation of the node.

[1]Perceptron and neuron are interchangeable terms.

2.6.2 Multi-layer Perceptrons

Activities in the field of neural networks came to stagnation after Minsky and Papert [71] showed the limitation of a single-layer perceptron of solving only linearly separable problems. A famous example of perceptron's limitation is the XOR problem which a single-layer perceptron is not capable of its simulation [72]. This was where the need for a multi-layer neural network to learn more complicated systems arose. As Figure 2.13 shows, a fully connected multi-layer neural network is built upon the connection of multiple layers of perceptrons, where the first layer is the input layer, directly receiving the data, the last layer is the output layer and all other layers between the input and output layers are referred to as hidden layers. The terms *width* and *depth* also indicate the number of neurons in a layer and the total number of layers, respectively.

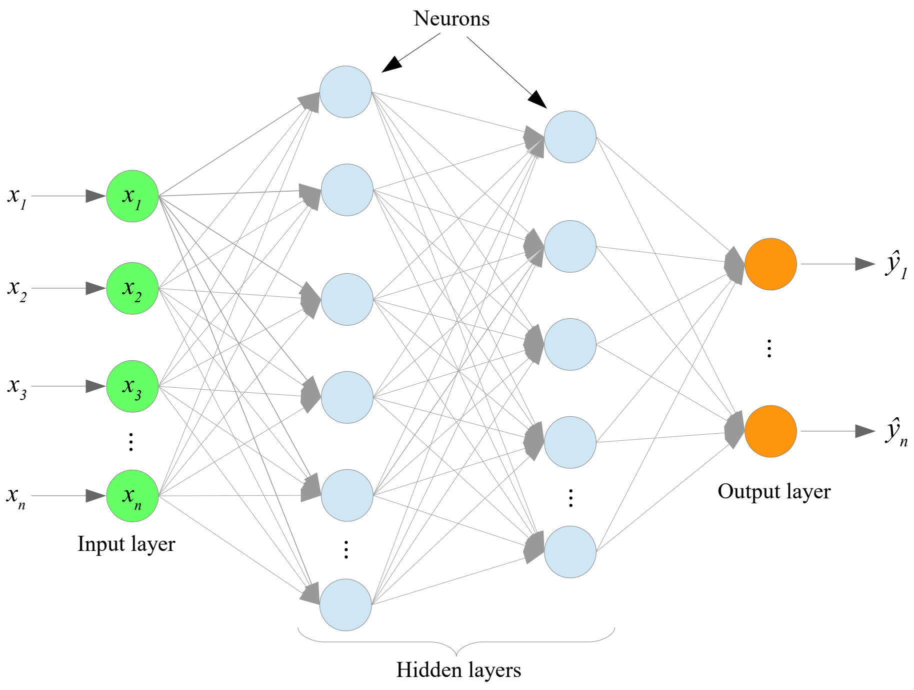


Figure 2.13: Multi-layer neural network

2.6.3 Activation Functions

In a neural network, the purpose of activation function f (also known as transfer function) is to rescale the amplitude of the output of a neuron between particular values and ensure the nonlinearity of the signal before feeding to the succeeding neurons in the next layer [73, 74]. This function has several forms, of which some of popular forms are presented below.

- **ReLU**: Due to low computational effort, rectified linear unit, or short, ReLU is the most popular activation function in ANN. ReLU limits the neuron's output between zero and values greater than zero by considering zero for all subzero signals and returning the values ≥ 0 with their own magnitudes as

$$f(z) = \begin{cases} 0 & \text{if } z < 0, \\ z & \text{if } z \geq 0, \end{cases} \tag{2.17}$$

and the corresponding graph showing this function is illustrated in Figure 2.14.

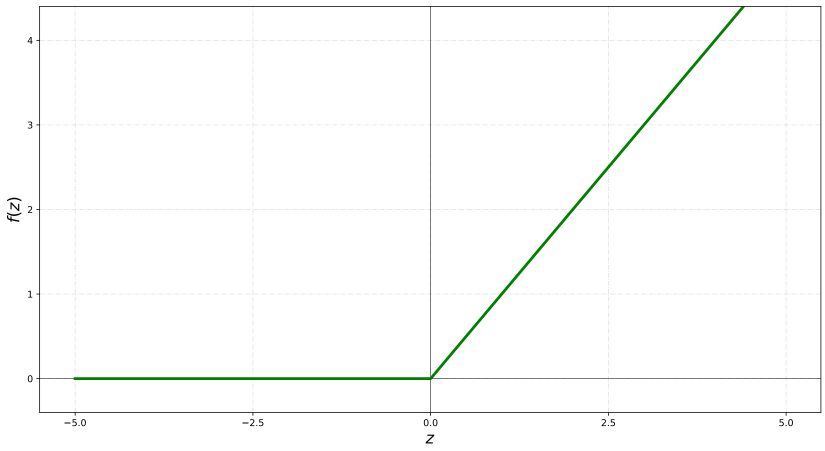


Figure 2.14: ReLU activation function

- **Sigmoid**: As shown in Figure 2.15, the Sigmoid is a nonlinear function and is regarded as an extensively used activation function in ANN. Sigmoid (also referred to as logistic or Soft step) transfers the values to a range of 1 and 0, and defined as

$$f(z) = \frac{1}{1+e^{-z}}. \tag{2.18}$$

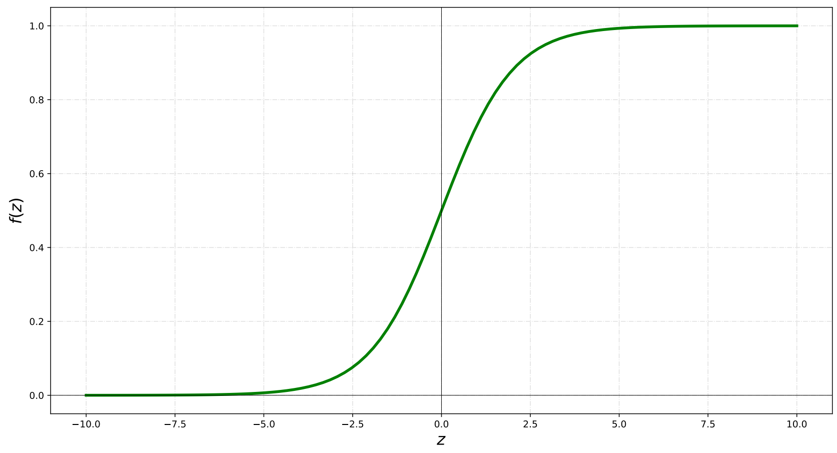


Figure 2.15: Sigmoid activation function

Sigmoid is a computationally demanding activation function due to its exponential form.

- **TanH**: As Figure 2.16 also demonstrates, a hyperbolic tangent function is in shape similar to the Sigmoid activation function, while it rescales the summed value of the node within -1 and +1. Hence, the neurons situated inside the next layer are not constantly receiving values of the same sign. Tanh is defined as

$$f(z) = tanh(z) = \frac{2}{1+e^{-2z}} - 1. \tag{2.19}$$

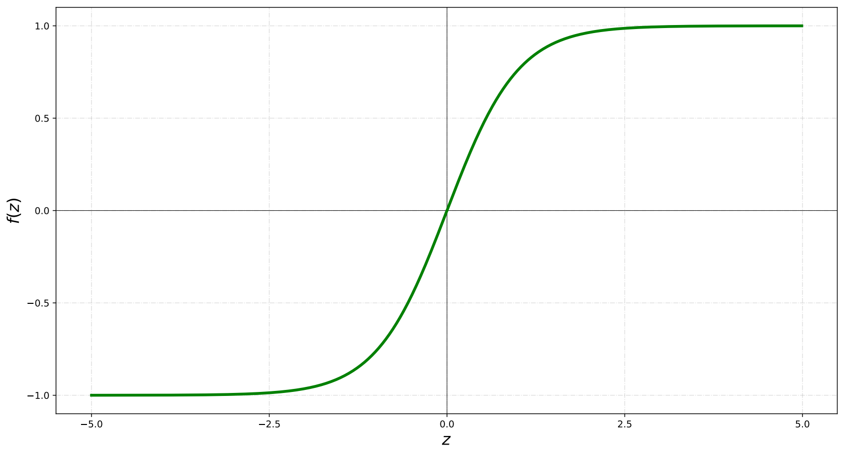


Figure 2.16: Hyperbolic tangent activation function

In contrast to the Sigmoid activation function, TanH facilitates the subsequent layer's learning since the mean value of the function is close to zero.

- **Softmax**: This function is typically used in the final layer in classification problems with multiple classes. The Softmax function converts all the outputs into normalized probabilities distribution of the target class over K possible target classes and is written as

$$f(\vec{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{2.20}$$

Thus, if the vector of the outputs $\vec{z}$ enters a Softmax activation function, the probability of each class will be as shown in Figure 2.17.

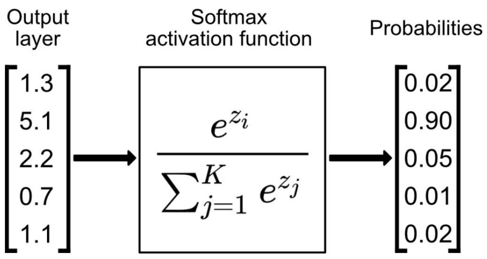
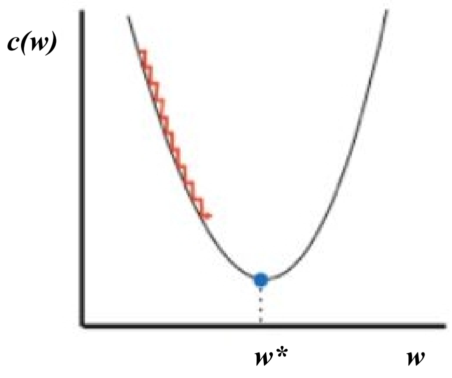

Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

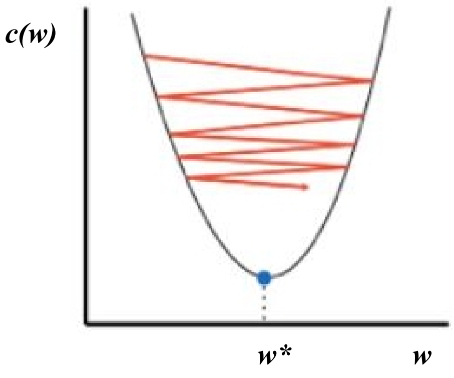
Figure 2.17: Softmax activation function [75]

2.6.4 Cost Functions

Cost functions (also known as loss or error functions) are used to compare the neural network outputs with the actual values in supervised learning; i.e., they are monitoring the network's performance by returning a value that is subsequently used for updating weights and biases through a process known as *backpropagation*. That being said, during each iteration of training, the goal is to reach convergence to a minimum cost. A cost function in a feedforward neural network is a function of weights w, biases b, the input of a training example S^r, and the expected output of S^r, E^r. Therefore, if the cost function is only a function of a weight or a bias, the cost function can be minimized by finding optimal values of w and b through an approach known as *gradient descent* which is shown in Figure 2.18, where the steps sizes are referred to as *learning rate* [76].



(a) Too small steps leads to slow convergence



(b) Too large steps can result in overshooting and divergence

Figure 2.18: Gradient descent [77]

Three popular qualified examples of cost functions to use in regression tasks are Mean-Squared Logarithmic, Mean Absolute Error (MAE) and Mean Squared Error (MSE) and respectively defined as

$$C = \frac{1}{n}\sum_{i=1}^{n}(log(y_i + 1) - log(\hat{y}_i + 1))^2, \tag{2.21}$$

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i), \tag{2.22}$$

and

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.23}$$

where $\hat{y}_i$ is the predicted output for a point, y_i is the true value of the same point and n is the total number of points.

2.6.5 Optimization Algorithms

To minimize the loss during training, certain values of the network parameters such as learning rate and weights need to be found. Optimization algorithms define the extent and manner of changes in these parameters and regulate the path through finding the cost function minimum values. In other words, the approach by which the NN learns is defined by optimizers. In deep learning, several optimization algorithms exist. However, in this subsection, only the most well-known algorithms suitable to use in regression tasks are briefly presented, and for further information, the reader is encouraged to refer to highly educative online sources such as Deep Learning Demystified [78].

Stochastic Gradient Descent

Stochastic gradient descent, or in short SGD, is an efficient variant of gradient descent with the difference of random selection of a sample data point instead of performing derivation calculation for the entire dataset. This results in a significant reduction in computational overload associated with using normal gradient descent algorithm [79].

Adam

This optimization algorithm has a solid reputation for reducing the processing effort needed to find the cost function's global minima. Adam takes advantage of an adaptive learning rate algorithm by opting for different step sizes for different parameters that can lead to faster convergence when noticeable updates in the weights do not occur [80].

As examples for other well-known gradient-descent-based optimization algorithms, RMSprop, Momentum, Adagrad, and Nadam can be mentioned [81]. However, their explanations are out of the scope of this thesis.

2.6.6 Backpropagation

An artificial neural network is comparable to a function capable of taking any shape through choosing the coefficients of the function. This flexibility of ANN can be reached by proper adjustment and optimization of weights and biases of the network (training parameters) during the training phase. Having a training dataset with the input and known outputs available, the predicted values of the final layer are evaluated against the expected results using the loss function. The deviation from the training outputs can then be minimized by going backward in the network and adjusting the training parameters to reach an acceptable range of error. This process is referred to as backpropagation [8, 82].

2.6.7 Generalization and Overfitting

A well-trained artificial neural network is expected to generalize the input-output relationship even for the datasets with slight differences from those used to train the network. For doing so, the network must see many training instances to be able to estimate the corresponding outputs of unseen inputs. By seeing the network as nonlinear mapping of the input signal to the outputs, a curve fitting approach in which generalization for new points accomplished by interpolation can be a good example for understanding the network's overall function. Nevertheless, excessive data points or the presence of noise in the training data might eventuate in a phenomenon known as *overfitting* or *overtraining*, where the network memorizes the data instead of modeling the underlying function present in the dataset. Therefore, an overtrained network is not capable of proper generalization of the input-output relationship [70]. As opposed to overfitting, the term *underfitting* exists, where the neural network is not trained with enough data. Thus, the accurate prediction of an undertrained network for new inputs is under question. Figure 2.19 illustrates the concepts of underfitting, overfitting, and balanced fitting.

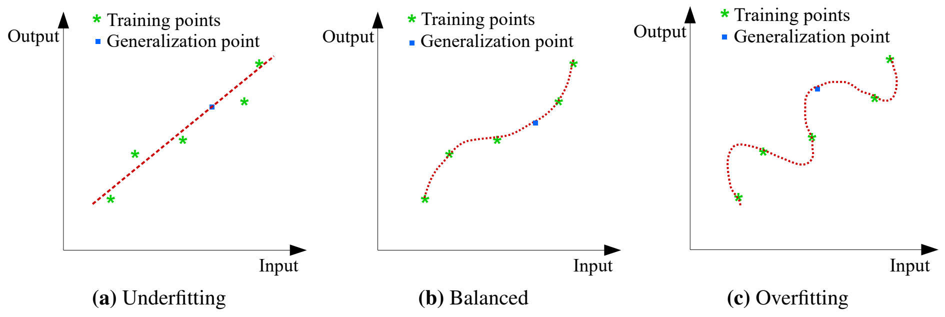


Figure 2.19: An illustration over the concepts of overfitting, balanced, and underfitting

Chapter 3

State-of-the-art Review of the Employment of Machine Learning and Finite Element Analysis in Biomechanics of Soft Tissues

Machine learning techniques as a high-speed substitution for FEA simulations in biomechanical applications have been of high interest in the literature and increased in the past decade. Many researchers have implemented different ML algorithms for the prediction of large deformation in various organs and provided their comparative results, emphasizing the advantages of using particular algorithms and constitutive laws over the others. This chapter of the thesis highlights the most recent techniques used for simulating the behavior of biological organs, such as the 3D reconstruction of geometries, applied constitutive laws, FEA considerations, ML models, accuracy, and computational costs, as a way for better understanding of progress expanse in this field.

3.1 Liver

The framework presented by Morooka et al. [83] in 2008, to our best knowledge, was the pioneer of the researches exploring a solution to reduce the response delay of FEM simulations of the liver by resorting to a neural network. The method proposed in this work was based on the superposition of basic deformation modes of the model where the network learned the organ's response to the applied external forces. The approach adopted in this work consisted of these general steps:

- discritization of the computational domain using hexahedral elements

- generation of 500 pairs of external forces with random directions and apply them as nodal loads
- capturing the resultant nonlinear deformation modes caused by these forces
- building the required dataset for the training of the neural network
- setting up the neural network with empirically determined numbers for width and depth of the network
- training the network with the dataset generated from FEM and using a backpropagation algorithm to limit the error below the defined threshold

While this study does not provide any information regarding the utilized hyperelastic model, including its parameters and the model order, accounting for the effect of vascularization or the stiffness of the Glisson's capsule, which are essential for a physically realistic simulation, it stated that, on a PC platform of Pentium4 2.8 GHz with 1 GB memory, the neural network model expedited the computational time by a factor of 1000 faster than nonlinear FEM and produced the outputs in 0.28 ms.

Another work investigating the integration of data-driven models in computer-aided surgery of the human liver was presented by Lorente et al. [49]. This study evaluated the feasibility of using ML for real-time simulation of the liver's biomechanical behavior in the course of the inhalation and exhalation processes. For this purpose, three tree-based regression algorithms, namely, decision tree (DT), random forests (RF), and extremely randomized trees (ET), and two other comparatively less complicated methods of dummy model (DM) and linear regression (LR) were evaluated. The goal of this research was to develop a scheme not only capable of capturing large hyperelastic deformations, but to have the ability of prediction for different geometries of the liver. Thus, this study used data from eight geometrically different livers to train and test the ML regression models.

The first step in this work was to reconstruct the models suitable for FEM by performing Computed Tomography (CT) scanning of the livers. The CT images were then segmented, and voxel-based meshes were acquired. The second-order Ogden material model was selected, and a combination of three different material parameters was used to create a richer dataset. The model was then discretized with tetrahedral elements, and simplified boundary conditions, comparable to the interaction of the liver with different abdominal organs during breathing, were defined. After running the simulations, the data describing the deformed state of each liver in the 3D Euclidean space were gathered. These data included the initial coordinates of the nodes, nodal displacements, and applied nodal loads, in the later stages, together with the material parameters constituted to the required dataset for the training of the ML models.

For splitting strategy, two approaches were adopted. The first strategy was to leave one liver for testing and the seven other livers for the training of the models. The second approach was 70$\%$ of the total samples (2,208,486 samples) obtained from the FEM simulation were used to train the models, while the remaining 30% (946,494 samples) was allocated as a test set. The hyperparameters, which are not automatically learned inside the ML models, were optimized for the tree-based models. Hyperparameter optimization ensures the generalization of the model and prevents overfitting and underfitting the training dataset, which subsequently increases prediction performance. After training the models, the models' performances were evaluated using root mean squared error (RMSE) which is

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{test}} (\hat{y_i} - y_i)^2}{n_{test}}}, \tag{3.1}$$

and n_{test} defines the number of samples in the test set. Moreover, another metric known as Mean Euclidean Error (Eq. 3.2), capable of calculating the actual nodal displacement shown in Figure 3.1, was used to measure the percentage of samples with displacement error equal to and lower than 1 and 3 mm.

$$d(\hat{y_i}, y_i) = \sqrt{(\hat{dx_i} - dx_i)^2 + (\hat{dy_i} - dy_i)^2 + (\hat{dz_i} - dz_i)^2} \tag{3.2}$$

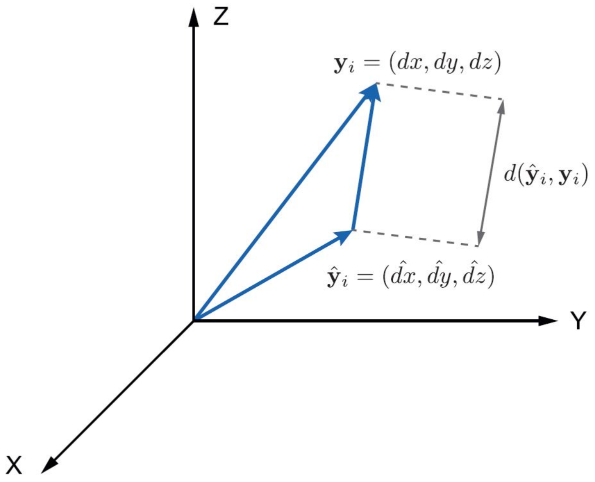

Z
$\mathbf{y}_i = (dx, dy, dz)$
$d(\hat{\mathbf{y}}_i, \mathbf{y}_i)$
$\hat{\mathbf{y}}_i = (\hat{dx}, \hat{dy}, \hat{dz})$
Y
X

Figure 3.1: Displacement error in Euclidean space [49]

As seen in Table 3.1, the comparative performance results for the five ML models are presented, where the best results have been highlighted in bold. This table indicates the performance superiority of the random forests algorithm in the first data splitting strategy. At the same time, this algorithm was in second place in percentage splitting with slightly less accurate prediction than the extremely randomized trees model. It is also noticeable that the tree-based models outperformed the dummy model and linear regression. In terms of computation cost, this work compared the time consumed by FEM against a trained RF regression model for the same loading scenario. The simulations were conducted on a computer based on a 3.4 GHz Intel Core i7 CPU and 8GB RAM. ML model completed the task in 2.89 seconds, while it took FEM 51.63 seconds to finish the simulation. It is also worth mentioning that the computational time of 2.89 seconds is associated with the simulation of ten deformed states, while in medical applications, the prediction of only one liver is required. Therefore, a value of 0.3 seconds can satisfy real-time criteria.

Table 3.1: Performance assessment of ML models used in the research work of Lorente et al. [49]

Regression model	One liver for test: Samples with Euclidean errors ≤1 mm (%)	One liver for test: Samples with Euclidean errors ≤3 mm (%)	One liver for test: Mean Euclidean error (mm)	70%/30%: Samples with Euclidean errors ≤1 mm (%)	70%/30%: Samples with Euclidean errors ≤3 mm (%)	70%/30%: Mean Euclidean error (mm)
DM	2.20	22.87	5.41	2.31	21.23	5.41
LR	15.40	66.67	3.04	10.94	59.62	3.22
DT	57.77	93.20	1.13	99.66	100.00	0.14
RF	**60.19**	**94.31**	**1.06**	99.99	100.00	0.09
ET	55.45	91.67	1.20	**100.00**	**100.00**	**0.07**

Pellicer-Valero et al. [6] also published their research in late 2019, employing feedforward neural network and random forests algorithms to build models trained by data from FEM simulation of more than 100 different shapes of the liver. The liver segmentation needed for the simulations was conducted in a fully automatic process, and tetrahedral elements were used for discretization. First-order Ogden was employed as the hyperelastic model, and for the boundary conditions, the nodes in contact with inferior vena cava were considered fixed with null displacement. Nodal forces with random orientation and constant magnitude of $0.4N$ were applied, and the corresponding nodal displacements were obtained in the post-processing stage. For maximizing the richness of the dataset, four different scenarios shown in Table 3.2 were selected. These scenarios were considered to account for the cases of the presence of two surgical tools, material properties change due to, e.g., inflammation, and the fact that a liver does not possess a unique shape. By omitting viscoelasticity, the static analyses were performed. A dataset including information about the nodes' initial coordinates, force vector components, coordinates of the nodes after displacement, and material properties was created to train the ML models.

Table 3.2: Simulation scenarios in the work presented by Pellicer-Valero et al. [6]

ID	Two forces	Multiple materials	Multiple geometries	Simulated forces
1	No	No	No	400
2	Yes	No	No	1500 ($\times$2)
3	No	Yes	No	3,000
4	No	No	Yes	10,200

As for validation of the neural network performance, 10% of the simulations were set for the first three scenarios, while for the last scenario, this number was 12.5%. ReLU activation function was also employed, and the network's performance was monitored using the MSE cost function. The performance of the neural network for all scenarios is presented in Table 3.3 based on Mean Absolute Error, Mean Euclidean Error, and the percentage of samples with Euclidean error below 1 and 3 millimeters.

Table 3.3: Neural network performance measurement for Scenarios in work presented by Pellicer-Valero et al.

Scenario	MAE (mm)			Mean Euc. Error (mm)	% of samples	
	x	y	z		E_{Euc} 1 mm	E_{Euc} 3 mm
1 (base case)	0.1173	0.1176	0.1224	0.2389	98.2615	99.9551
2 (two forces)	0.1862	0.1825	0.1977	0.3816	93.0551	99.8382
3 (30 materials)	0.1621	0.1616	0.1548	0.3299	94.7381	98.4644
4 (102 livers)	0.4130	0.4732	0.4119	0.8643	72.4243	95.5784

Regarding the real-time applicability of this work, this work tested the neural network response time by resorting to GPU computational power on a computer equipped with two Core i5 processors and a low-end GT 840M GPU. For the first and last scenario, the computational times measured to be $2ms$ and $5ms$, respectively. Hence, using superior hardware, the required working frequency of a haptic feedback system, which is above $500Hz$, could be achieved.

The other research works contributing to the implementation of machine learning to the real-time simulation of the deformed liver are Pfeiffer et al. [84], and Mendizabal et al. [7]. These works used fully convolutional neural network based on U-Net architecture as the machine learning algorithm to approximate nonlinear deformation of the liver. They showed that their methods were high-speed with high accuracy and capable of handling real-time conditions. With this regard, the model proposed by Mendizabal et al. [7] was able to predict the organ deformation in $3ms$ with the maximal error below 0.4 mm. The training time, however, was significant, with 149 min in value.

3.2 Brain

The study conducted by Tonutti et al. [10] in 2017 aimed at the localization of tumors during neurosurgery. This work proposed an approach combining the results of FEM simulations with artificial neural network and support vector regression (SVR) algorithms. The general workflow of this approach is outlined in Figure 3.2, where the patient-specific mesh was obtained from performing pre-operative magnetic resonance imaging (MRI) scans and high-quality segmentation of brain matters. The first-order Ogden was the constitutive model of choice, and the density and bulk modulus of the tumor were set twice as those of the healthy brain tissue to simulate its higher stiffness. For the application of loading, 11 nodes were considered to receive a force with respectively 30 and 20 different orientations and magnitudes (from $0.1N$ up to $1N$), resulting in 600 simulations per node and 6600 FEM simulations in total. Furthermore, no-slip boundary conditions were decided for the majority of the nodes in contact with the internal surface of the skull, which is illustrated as unshaded wireframe in Figure 3.3.

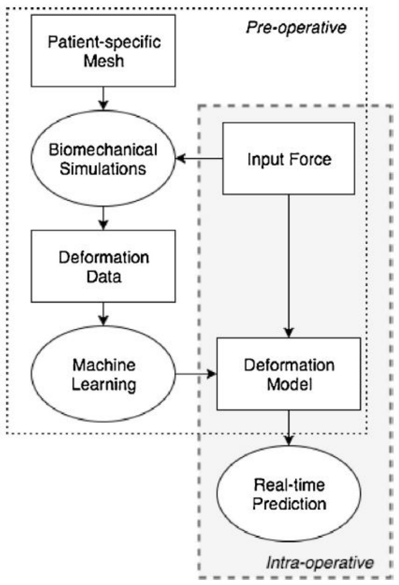


Figure 3.2: Outline of Tonutti et al. work [10]

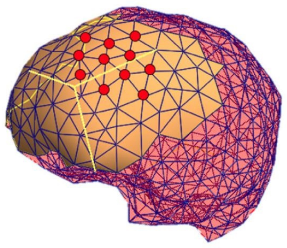


Figure 3.3: BCs used in Tonutti et al. work [10]

In the later stage of this work, a neural network with one hidden layer of four neurons was trained based on 70% of the data. Bayesian regularization was applied to the inputs to lower the possibility of overfitting. The authors reported that the ANN model predicted the tumor nodes' position by providing an accuracy below 0.3 mm, whereas the SVR model performed better with an error under 0.2 mm. Both values, however, were stated to lie within the acceptable threshold for this application.

3.3 Breast

In the area of the application of data-driven methods combined with FEM for characterizing biomechanical behavior of human breast tissue, the study performed by Martínez et al. [11] stands out in the literature. This work took advantage of the data generated from FE analysis of ten different shapes of the breast for the training of three machine learning models in the hope of developing a framework for the real-time application of image-guided surgery for tumor tracking.

The entire workflow followed in this work, where the segmented models were used for tetrahedral-discretized meshes required in FEA, is outlined in Figure 3.4. As for the constitutive law, Mooney-Rivlin hyperelastic model was utilized for three different tissues, namely glandular tissue, fat, and skin, existing in the breast. Regarding the load and boundary conditions, the breast was considered to be compressed between two plates by 20% of its height, which imitates the real-world scenario during medical intervention. The simulations were run in several load steps, automatically chosen by FEM software, resulting in 162 deformation modes for each breast. This led to 9,816,283 instances constituting the dataset used in the training of the machine learning models.

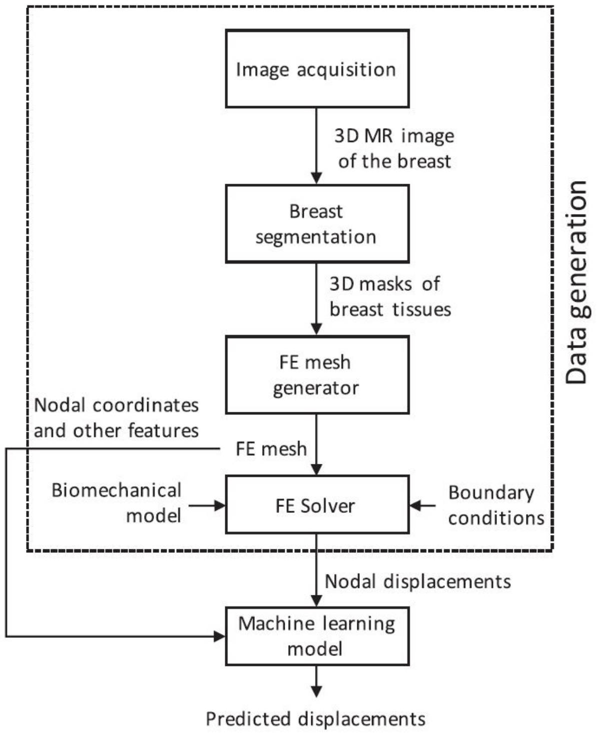
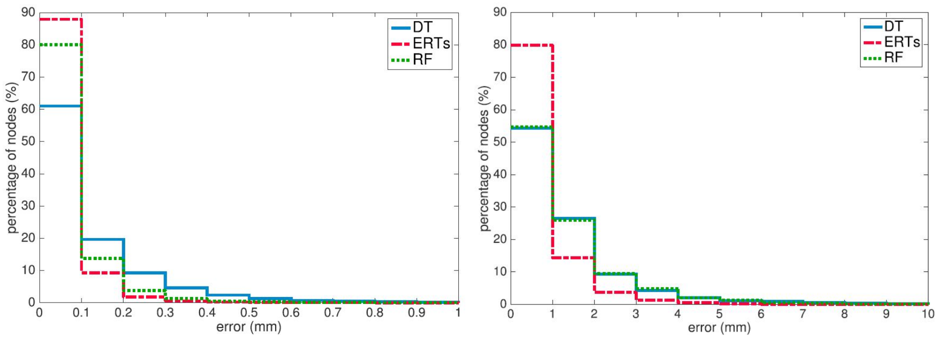


Figure 3.4: Outline of Martínez et al. proposed workflow [11]

The authors evaluated the performance of random forest, decision tree, and extremely randomized trees (ERTs) as machine learning algorithms. Two approaches were also adopted as partitioning strategies. The first method (hold-out) was a random division of the instances in the entire dataset based on 70% for training and the remainder as the validation set. In the second approach (leave-one-deformation-out), the instances related to one of the 162 deformation modes were used to test the models' performance. This process repeated 162 times with all deformations.

Once the training of the models was done, the error of prediction pertaining to each node was evaluated using 3D Euclidean distance. As the distribution of these errors shown in Figure 3.5 indicates, in the two splitting approaches, ERTs performed the best by having the smallest percentage of significant errors among the other two ML models. Nevertheless, in the hold-out approach, better performance of the models was obtained. The time required by the best model to predict the deformation was also reported to be less than 0.2 seconds, showing the capacity of the model to be implemented for real-time predictions.



(a) Hold-out **(b)** Leave-one-deformation-out

Figure 3.5: Error distribution of splitting strategies in the work of Martínez et al. [11]

Chapter 4

Methodology And Simulations

To fulfill the objective of this thesis, several steps were taken, and several computer tools, each serving specific purposes, were utilized. This chapter intends to present the undertaken approaches and introduce the tools and techniques used in this work. For this reason, it is essential to have a general overview of the required steps based on the information presented in the previous chapter. This can ascertain the logical implementation of the methods and flow of the information. The overall flowchart of the identified works, expected to be carried out in this project, is shown in Figure 4.1. The following subsections aim at outlining the steps and technical aspects that existed in the project.

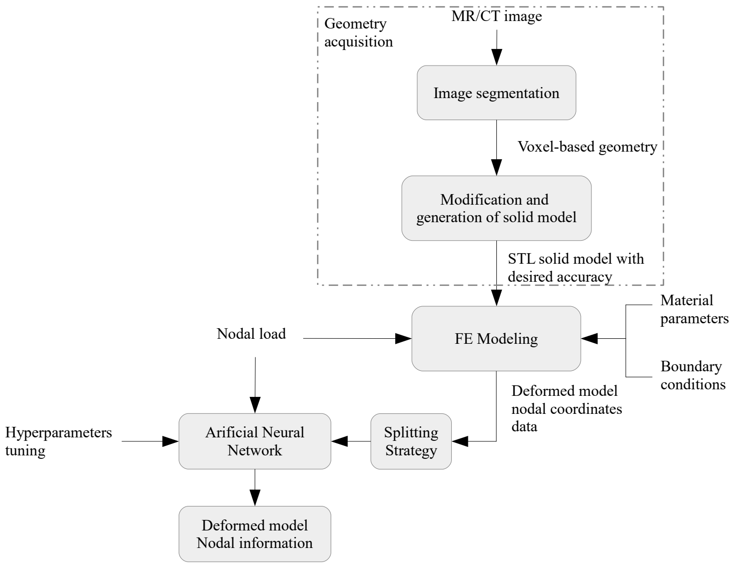


Figure 4.1: The flowchart of proposed method

4.1 Preparatory Works

Prior to performing FE analyses, some preliminary tasks were performed to identify the key features and parameters useful to extract for training ML models. Since working on a liver with all its geometrical complexity in 3D seemed challenging, the initial approach was first to use a simple geometry for performing FEM simulations. Therefore, FE analysis of a 2D hyperelastic rectangle was considered to generate data necessary for training a simple ML model (random forests) to lay the foundation for the future actual-shaped model. This method resulted in two main benefits. First, the viability of the project was checked in an early stage. Then the most challenging part of the simulations, which was automatization of the processes, to a large degree, was solved and understood. In the later sections, the details of the works conducted on the actual organ geometry are presented.

4.2 Geometry Acquisition

The first step towards building geometry of liver is to obtain the scanned image of the abdominal area of the patient. This image can be either obtained through CT scanning or MR imaging. Both techniques can provide a multi-layered view of the body's internal structure, including skeletal structure, blood vessels, and organs soft tissues. CT scanning, however, is quicker to perform, while MR images provide better resolution and contrast, which is beneficial for differentiation of various organs, matters, and abnormal tissues like tumors [85]. Regardless of the superior quality of MR images, this work used an abdominal CT image of an anonymous patient provided by the Netherlands Cancer Institute.

4.2.1 Segmentation

The CT images from the source mentioned above contained information about all the tissues in the patient's abdominal area. Therefore, a step known as *segmentation* was required to be carried out in order to isolate the organ/tissue of interest. Today, due to the recent development in computer vision and the power of convolutional neural networks, many researchers [86–88] have managed to partially or even fully automatize the process of segmentation. These techniques significantly expedite the process of acquisition of 3D data. Nevertheless, due to the technical complexity of the process and time limitation, a manual approach using the open-source software 3D Slicer [89] was selected.

As Figure 4.2 shows the viewports of the software 3D Slicer, the geometry of the liver was reconstructed from a CT image. The basic principle of manual segmentation used in this thesis was based on the determination of the volume and boundary of the organ as seen in different layers of the CT image. The software could then approximate the void space between the layers to create a voxel-based[1] geometry. The higher the number of layers used in segmentation is, the more accurate the organ shape will be. However, the resulting

[1]In a 3D model on a regular grid, a voxel is the smallest constituent unit making up to the totality of the object. A voxel is comparable to a pixel on a 2D image.

geometry still included noise and artifacts that made it unsuitable for a FE analysis.

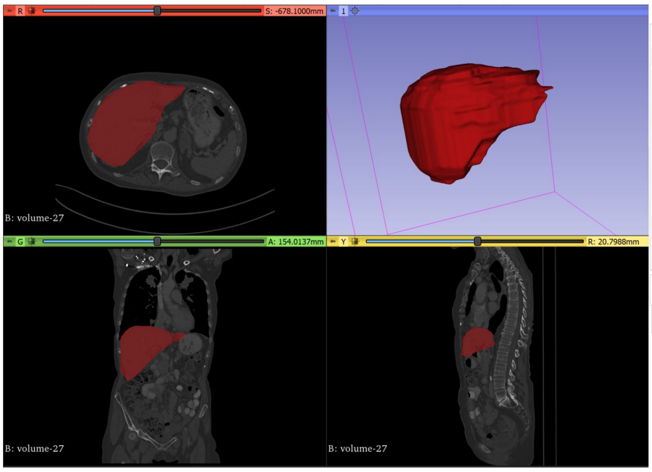


Figure 4.2: Abdominal CT image and segmented liver

4.2.2 Modification and Generation of Solid Models

The model created in 3d Slicer environment was voxel-based, had noises and unwanted features, and, more importantly, lacked the characteristics of a solid model such as volume that makes it suitable to use in 3D FE analysis. Hence, Autodesk Meshmixer [90], another intermediary free software package capable of delivering a suitable model for meshing, was utilized, and the liver model was imported into the Meshmixer in stereolithography (STL) file format. As seen from Figure 4.3, the model's topology was reconfigured, and extra features and noises were discarded. Moreover, the number of surface triangles was adjusted to a reasonable value to preserve the shape of the organ without adding up to the computational time.

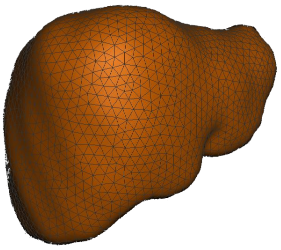


Figure 4.3: Modified liver geometry and conversion of voxels to standard triangles

4.3 Finite Element Simulations

The finite element analyses conducted in this project dealt with high material nonlinearity which is the natural characteristic of a liver under loading conditions. The Abaqus package has a well-recognized ability to handle nonlinear simulations. Plus, the standard hyperelastic models such as Ogden, Mooney-Rivlin, Yeoh have already been implemented in the software. Another significant advantage of the software is that the entire FE simulation from pre-processing steps, including meshing, loading, material assignment, and post-processing, is programmable with Python language. This capability not only helped to avoid repetitive tasks for performing a large number of simulations required for training of the neural network, but it also provided flexibility in the choice of loading conditions imposed on the organ.

The initial approach taken was to see the liver as a homogeneous object without considering the effect of hepatic vessels to lower the computational time. The lack of a proper segmentation mask also led to disregard the effect of the Glisson's capsule. Viscoelasticity would also be diminished by applying loads in small increments and avoiding any dynamic situation. However, the intention was to later embed the vessels in the simulations, which is of significant importance during hepatic surgery of the liver.

4.3.1 Material Model and Parameters

As covered in Section 2.3, several hyperelastic models were proposed and used in previous works. Nonetheless, in the literature, usage of first and second order of Ogden model is prevailing to describe the liver's parenchyma. This model has also been reported to provide the best fit with the material's stress-strain curve [14, 49].

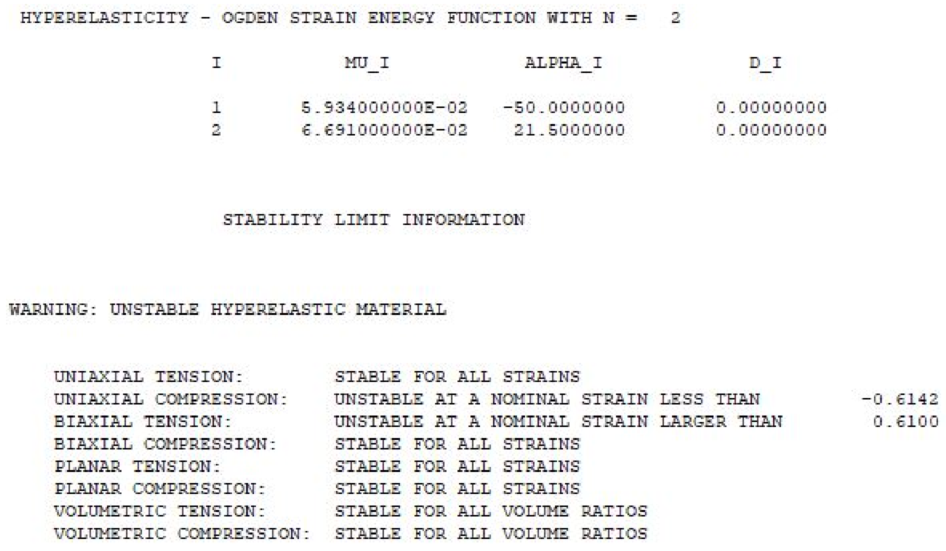
Material Evaluation

Four combinations of material parameters were obtained from the work of Lorente et al. [49], and Pellicer-Valero et al. [6]. As explained in Chapter 2, these parameters define the strain energy function of the material based on the Ogden formulation. The calculation of these parameters determines the correct shape of the function that fits best the experimental stress-strain curve from the test of liver tissue. Nevertheless, the constants obtained from the works mentioned above first needed to be evaluated for stability check. The Abaqus package is empowered to evaluate if the provided material parameters are stable for a specific strain regime. Table 4.1 shows the status of these combinations after evaluation.

Table 4.1: Ogden material parameters stability check

Combination of elastic constants	μ_1 (MPa)	α_1	μ_2 (MPa)	α_2	Stability status
1	0.06617	61.17	0.02203	98.75	✓
2	0.05934	-50	0.06691	21.5	✗
3	0.0119	-36.46	0.05767	99.7	✓
4	0.0041	10.06	N/A	N/A	✓

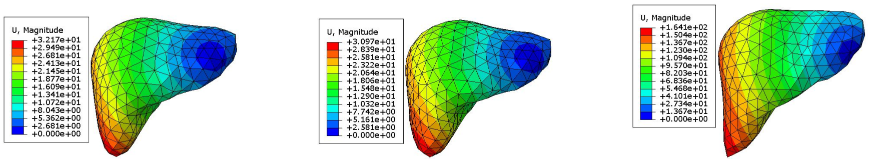
Figure 4.4 is a snapshot of the evaluation results, showing that the parameters presented as combination 2 in Table 4.1 was not stable for the entire strain regime in uniaxial compression and biaxial tension tests. Thus, this combination was discarded from the possible material parameters capable of adequately modeling the liver's hyperelastic behavior.



```
 HYPERELASTICITY - OGDEN STRAIN ENERGY FUNCTION WITH N =   2

                 I          MU_I            ALPHA_I             D_I

                 1      5.934000000E-02   -50.0000000        0.00000000
                 2      6.691000000E-02    21.5000000        0.00000000


                  STABILITY LIMIT INFORMATION


WARNING: UNSTABLE HYPERELASTIC MATERIAL


    UNIAXIAL TENSION:        STABLE FOR ALL STRAINS
    UNIAXIAL COMPRESSION:    UNSTABLE AT A NOMINAL STRAIN LESS THAN       -0.6142
    BIAXIAL TENSION:         UNSTABLE AT A NOMINAL STRAIN LARGER THAN      0.6100
    BIAXIAL COMPRESSION:     STABLE FOR ALL STRAINS
    PLANAR TENSION:          STABLE FOR ALL STRAINS
    PLANAR COMPRESSION:      STABLE FOR ALL STRAINS
    VOLUMETRIC TENSION:      STABLE FOR ALL VOLUME RATIOS
    VOLUMETRIC COMPRESSION:  STABLE FOR ALL VOLUME RATIOS
```

Figure 4.4: Ogden material parameters stability status

The remaining three combinations were also compared in terms of their resulting deflection under a 10 N nodal force, applied at the lowest node of the right lobe. As is shown in Figure 4.5, the first and third combinations presented by Lorente et al. [49] demonstrated higher stiffness with the maximum displacement magnitude of slightly above 3 mm, whereas, the fourth material parameters proposed in the work of Pellicer-Valero et al. [6] resulted in the significant value of 164 mm as the maximum displacement.



(a) Combination 1 **(b)** Combination 3 **(c)** Combination 4

Figure 4.5: The effect of combination of material parameters on the deflection results in the form of contour plot

Ultimately, the first combination was selected for proceeding with the FE simulations of this stage. The corresponding stress-strain curve of this combination is shown in Figure 4.6.

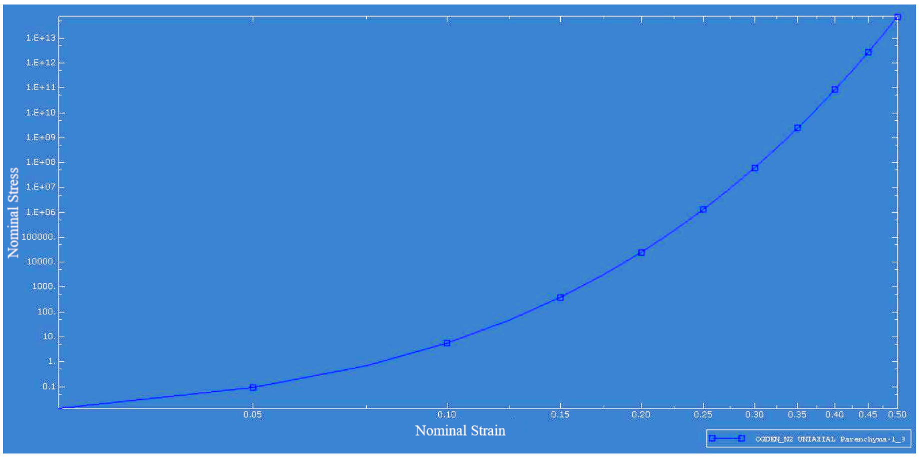


Figure 4.6: Stress-strain curve of the selected material parameters

4.3.2 Meshing

The surface of the model obtained from the Section 4.2.2 step was constructed by a finite number of triangles. Therefore, it was logical to select an element that would not cause a major change to the representation of the model's surface. Unlike hexahedral, tetrahedral elements are known for a better approximation of complex geometries. Since the surface of the organ was already built by triangles, this type of element was selected. The element of interest in the Abaqus/Standard package exists with C3D4H code, which stands for a continuum three-dimensional solid element with four nodes and a hybrid formulation. The hybrid formulation is used for a material with near or fully incompressible behavior whose response, except in the case of plane stress, cannot be modeled with traditional elements [33]. It must also be noted that in Abaqus, reduced integration cannot be used for tetrahedral elements and these elements only use full integration.

Furthermore, the maximum and minimum size of element edges were adjusted to be adaptive with triangles already existing in the model. The reason for adopting this approach was to prevent the over-division of the analytical faces. Thus, the total number of nodes for the definition of the overall geometry remained within a reasonable range, and the corresponding computational effort would not increase. Figure 4.7 demonstrates this approach for the basic geometry of the liver, where the location of the seeds overlapped with the corner of the triangles.

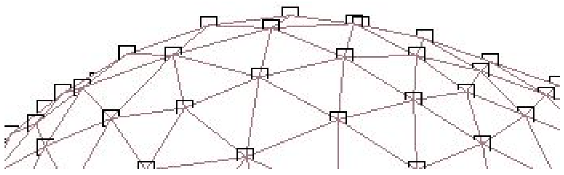
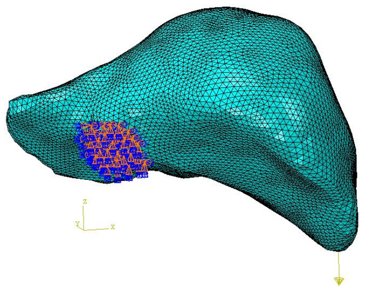


Figure 4.7: Mesh size control of the basic liver geometry

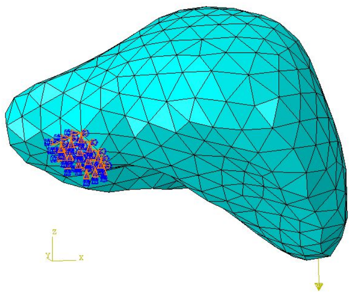
Mesh Sensitivity Analysis

The outer boundary of the liver extracted from the segmentation process can be considered geometrically complex and not capable of being defined by an analytical surface. This surface is instead formed by a collection of triangles representing an approximation of the entire organ. For acceleration of the FE simulations and regarding the large quantity of the simulations and storage capacity (both memory and hard disk) required for training of the neural network, it was preferred to lower the number of elements in the model as much as possible. Consequently, it was essential to investigate the effect of mesh refinement on the numerical results and, more specifically, on the magnitudes of the displacements given by the FE simulations.

Although the comparison of the influence of different meshes is usually made between models with similar boundaries, changing the mesh resolution also led to the change in the model's geometrical details. However, the overall dimensions and volume of the liver remained similar. For the mesh sensitivity study of the liver, two models were discretized with linear tetrahedral hybrid elements C3D4H, undergoing two loading cases, were considered. In the first case, as shown in Figure 4.8 a 10 N downward concentrated force was applied to a node with the lowest Z coordinate, while a force with the same magnitude in the negative direction of the X-axis was applied to the same node as for the second loading scenario.



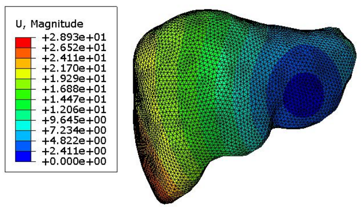
(a) Fine mesh



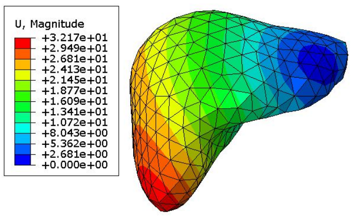
(b) Coarse mesh

Figure 4.8: Mesh sensitivity study first loading case with a 10 N concentrated force

The deformed shapes of the models under the first and second loading scenarios are respectively shown in Figure 4.9 and 4.10. As these figures illustrate, the resulting maximum displacement magnitude of FE analysis for the fine mesh was slightly smaller than the mesh with larger elements, with a difference of 0.324 mm. The difference in the second case was slightly larger with 0.364 mm. In both cases, however, the results were still reasonably close.

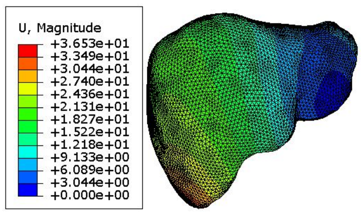


(a) Fine mesh

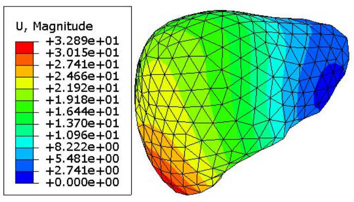


(b) Coarse mesh

Figure 4.9: Mesh sensitivity study displacement results for the first loading case



(a) Fine mesh



(b) Coarse mesh

Figure 4.10: Mesh sensitivity study displacement results for the second loading case

Another set of data that is of great significance is shown in Table 4.2, where mesh information, computational time, data sizes, and a summary of the resulting displacements are presented. As is clear, the usage of a refined mesh substantially increased the total CPU time and the size of stored data generated by Abaqus. At the same time, deformation results were generated with highly comparable accuracy. Therefore, using a coarse mesh for the upcoming simulations could be justified.

Table 4.2: Mesh details, computational time, data size and summary of displacement results, in mesh sensitivity analysis

Model	Number of elements	Elements edge size (mm)	Number of nodes	Total CPU time (sec)	Data size (MB)	Largest displacement (mm) (first case)	Largest displacement (mm) (second case)
Fine Mesh	206977	4 mm	38456	1030	1003	3.217	3.289
Coarse Mesh	3571	16 mm	797	6.8	14.1	2.893	3.653

4.3.3 Vascularization and Tumor

The network of blood vessels and bile ducts in the liver are geometrically complex yet play a crucial role in the response of the organ to external stimulants. Their locations are also of significant interest during surgical resection and tumor removal. Therefore, apart from the proof-of-concept objective of this thesis, one of the main goals was also the incorporation of these tissues in the FEM models.

Since the hepatic tree consisted of many intricate parts, the primary obstacle in the embedding of the vessels was obtaining a model, computationally affordable and representative of the geometry extracted from the segmentation process. Therefore, and as shown in Figure 4.11, the STL model from performed segmentation was reconstructed manually to reduce the complexity of the actual model.

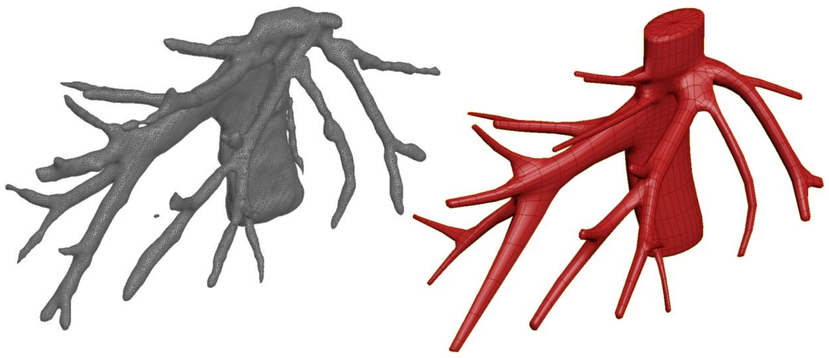


Figure 4.11: Approximation of hepatic vessels with simplified geometry

The simplified vessel structure was then first subtracted from the main liver tissue, and shell element S3 with the thickness of 2 mm was assigned to its faces in contact with the cavities in the liver. A sphere with a radius of 14 mm was also included in the FEM model to represent a tumor. The parenchyma, vessels, and tumor were then meshed simultaneously to ensure the conformity of their common faces. The material parameters used for the vessels belonged to the porcine liver and were Ogden-based, with $\mu = 0.0196\ MPa$ and $\alpha = 10.3043$ extracted from the work of Umale et al. [91]. As for the tumor, due to the lack of reliable data for the mechanical properties of cancerous tissues, the same parameters as those of the parenchyma were used. However, in general, a tumor is known to stiffer than its surrounding tissues in the liver. In Figure 4.12, the FE model of the liver with hepatic structure and a tumor is shown in wireframe display mode.

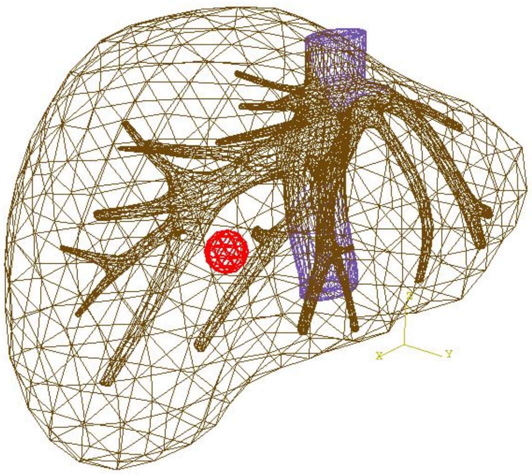


Figure 4.12: Liver model with hepatic vessels and tumor

Furthermore, the simplified model of the vessels was compared to the actual one by performing a FE analysis under the same loading, boundary condition, and material properties. As Table 4.3 indicates, the deformation results, which is the primary quantity of interest in this thesis, remained close in both vessel structures.

Table 4.3: Comparison of FE simulation of models with segmented and approximated vessel structures

Model	Number of elements	Number of nodes	Simulation duration (sec)	Data size (MB)	Maximum Displacement (mm)
Segmented	286,378	59,807	714	626	11.97
Approximated	46,226	10,832	52	107	9.99

The complexity of the FE model in terms of the number of nodes reduced significantly when a reconstructed model of the vessels was used. This further resulted in a considerably lower computational time and storage need, which alleviated the demand for more powerful hardware. It is also worth mentioning that both simulations were executed in parallel processing with four cores on an Intel(R) Core i5-8265U CPU.

4.3.4 Boundary Conditions

The liver in the human body is in contact with several organs, making finding anatomically realistic BCs an arduous task. However, in the literature [6, 12], for the sake of simplification, two different locations of the liver have been considered as fixed with null displacement. In this regard, the most common practice has been to fix the nodes in contact or the proximity of the inferior vena cava. Another area that can also be included in the BCs is the points lying on the falciform ligament that separates the right and left lobes and is connected to the diaphragm and abdominal wall. Nonetheless, due to the lack of enough information in medical images for the segmentation and loss of details in meshing, finding the exact position of the falciform ligament is very challenging. Hence, as also shown in Figure 4.13, only the nodes in the proximity of vena cava were restricted in all degrees of freedom, and the remaining nodes were left free.

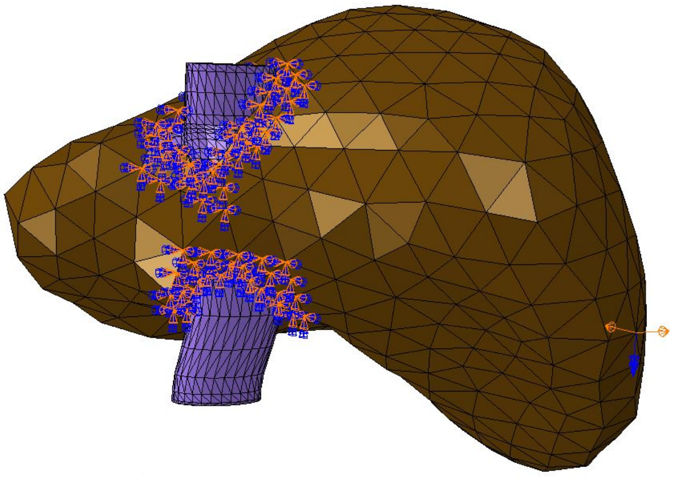


Figure 4.13: The boundary conditions of the liver

4.3.5 Loading and Simulations

Loading

Applying appropriate displacement loads to the FE model was the last step before running the simulations. In the case of this project, and as it was outlined in the introduction, the objective was to find the corresponding change in the location of the nodes for an external load applied on the surface of the liver. Therefore, the first step toward load application was to identify the nodes on the surface and randomly sample them with a defined percentage. The inclusion of a percentage of random load recipient nodes was decided for the following reasons:

- The quantity of the nodes on the surface was large, and full-loading of each required a massive amount of time, especially if a more refined mesh were to use. Lowering the simulations' duration is essential because of the time limit between medical imaging and surgical intervention.
- The location of the applied loads was not the only concern. The magnitude of the loads, their increment of application, and orientation were also important for producing a comprehensive dataset.
- Using all nodes on the surface could eventuate in saturation of the neural network model and subsequently compromising its generalization ability for unseen data.

Accordingly, 60 % of the surface nodes, equivalent to 109 nodes, were randomly sampled. As the Listing 4.1 shows the Python code for load generation, each sampled node was then applied displacement magnitudes in a range from 10 to and including 40 mm with the step size of 10 mm. For each magnitude, a total number of three different random orientations were also considered, and their components along the coordinate axes were stored in three lists D_x, D_y, and D_z, so that they could be consumed concurrently in a loop while performing the simulations. This led to 12 simulations per node, and by multiplying the number of simulation per node by the number of sampled nodes, the final number of simulations, 1308, was obtained.

```
import numpy as np
import math
F = np.arange(start ,stop ,step)
Dx, Dy, Dz, magnitude = [], [], [], []
for i in F:
    for j in range(3): # Here the total number of orientations per
magnitude is defined.
        phi=random.uniform(0, 2*math.pi)
        theta = random.uniform(0, 2*math.pi)
        x = round(i*math.cos(phi)*math.sin(theta),2)
        y = round(i*math.sin(phi)*math.sin(theta),2)
        z = round(i*math.cos(theta),2)
        Dx.append(x)
        Dy.append(y)
        Dz.append(z)
        magnitude.append(i)
```

Listing 4.1: Code used for generating random displacement loads

Simulations

The simulations were decided to be run on Abaqus/standard, which utilizes an implicit method for analyzing nonlinear problems. The static general step, which can handle linear, nonlinear, and quasi-static problems, was also chosen. This step does not account for the inertial effects and disregards time-dependencies in the material effects such as creep and viscoelasticity, yet it can be used for hyperelastic problems. Moreover, in nonlinear analyses, it is essential that the FEM solver divides each step into multiple time increments. Abaqus solver decides the time increment size based on

$$T = \sum_{i=1}^{N} \delta t_i, \tag{4.1}$$

where N is the limitation for maximum number of increments, δt_i is the increment size whose range is defined by $\delta t_{min} < \delta t_i < \delta t_{max}$. In FE analysis of hyperelastic materials, it is recommended to use small time increments to ensure the accuracy of the results [33]. Therefore, the maximum number of increments, initial, minimum, and maximum increment size were set to 1000, 0.01, 10^{-12}, and 0.1 respectively, which resulted in 15 increments, automatically chosen by the software.

Finally, the simulations were initiated using 12 cores CPU parallel processing on a cluster machine equipped with eight E7-8870 CPUs at 2.4GHz with a total number of 80 cores. The simulations could also be divided further to be processed on more than 12 cores. However, the higher number of cores did not necessarily result in faster computational time due to the time the software required to perform the jobs' division and assign them to each core. On average, the time consumed by the software to process each job was recorded to be 100 seconds, leading to approximately 36 hours for completing 1308 jobs.

4.3.6 Post Processing

In Abaqus, the results of FE simulations are stored inside output database (ODB) files. By running a Python script, these files were then called, and the field values requested in the pre-processing stage, such as nodes labels, nodal coordinates, and displacements, were extracted for each node and were written inside another file suitable for data analysis. For this purpose, it was decided to use comma-separated values (CSV) files. Therefore, for each simulation, there was a corresponding CSV file. Since the tumor and the network of hepatic vessels had shared nodes with the liver parenchyma model, it was only necessary to export the nodes' results on this tissue so that the location of other components could also be determined. This led to having 8388 instances in each CSV file.

4.4 Training of the ANN

4.4.1 Overview of the Tools

Although commercial software like MATLAB provides the essential toolbox for deep learning, the preference of this thesis was to maximize the utilization of open-source packages with a large community of users and developers. This ensured convenient access to learning materials and the availability of discussion forums. Moreover, in general, training a deep learning model requires high computational power that made using parallel processing a requirement to be met. For the aforementioned reasons, it was decided to use

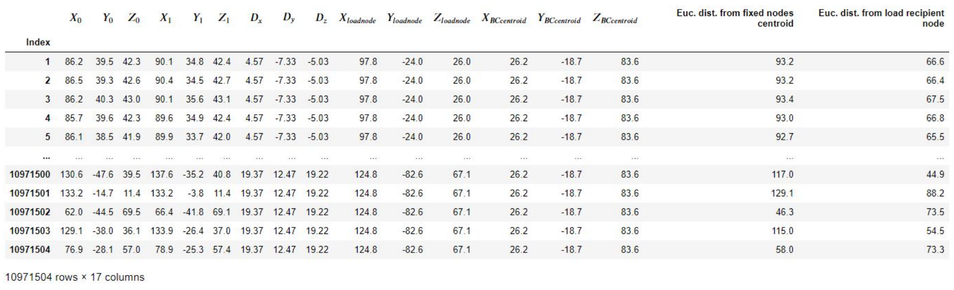
TensorFlow [92] which an open-source machine learning platform developed by Google Brain Team, and *Keras* [93] which is an open-source deep-learning interface running on top of TensorFlow as back-end library. The main benefit of the above choices for this work was the ease of implementing other Python libraries for data analysis and visualization, such as *Pandas* [94] and *Seaborn* [95] alongside the ML library in one coding environment.

4.4.2 Preparation of the Dataset

The 1308 CSV files obtained from the post-processing stage were then needed to be concatenated to build a unified dataset suitable to use for the preparatory steps before feeding to the machine learning algorithm. Therefore, they all imported into *Jupyter Notebook* [96], which is an open-source web application for live coding in Python, and stored in a dataframe containing 10,971,505 instances.

In order to provide enough information for the neural network algorithm to find the relationship inside the data, some other features were derived from the FEM data. The first three columns representing the X, Y, and Z initial coordinates of the nodes before deformation were added by subtracting their displacement values along axes of coordinates from their coordinates after deformation. Euclidean distances of the nodes from the centroid of the fixed nodes used in boundary condition was also added to the dataset as another feature. Finally, the Euclidean distances of the nodes from the nodes, which had received displacement loads in FE simulations, were also extracted as an additional feature. Table 4.4 shows the final Pandas DataFrame in the Jupyter Notebook environment. This DataFrame was built from the FEM information and contained 17 columns, out of which 14 were descriptive features. Since each row in the dataset was seen by the ML algorithm as an independent instance, it was crucial to provide the algorithm as much information as possible.

Table 4.4: The Pandas DataFrame prepared from FEM values



	X_0	Y_0	Z_0	X_1	Y_1	Z_1	D_x	D_y	D_z	$X_{loadnode}$	$Y_{loadnode}$	$Z_{loadnode}$	$X_{BCcentroid}$	$Y_{BCcentroid}$	$Z_{BCcentroid}$	Euc. dist. from fixed nodes centroid	Euc. dist. from load recipient node
Index																	
1	86.2	39.5	42.3	90.1	34.8	42.4	4.57	-7.33	-5.03	97.8	-24.0	26.0	26.2	-18.7	83.6	93.2	66.6
2	86.5	39.3	42.6	90.4	34.5	42.7	4.57	-7.33	-5.03	97.8	-24.0	26.0	26.2	-18.7	83.6	93.2	66.4
3	86.2	40.3	43.0	90.1	35.6	43.1	4.57	-7.33	-5.03	97.8	-24.0	26.0	26.2	-18.7	83.6	93.4	67.5
4	85.7	39.6	42.3	89.6	34.9	42.4	4.57	-7.33	-5.03	97.8	-24.0	26.0	26.2	-18.7	83.6	93.0	66.8
5	86.1	38.5	41.9	89.9	33.7	42.0	4.57	-7.33	-5.03	97.8	-24.0	26.0	26.2	-18.7	83.6	92.7	65.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10971500	130.6	-47.6	39.5	137.6	-35.2	40.8	19.37	12.47	19.22	124.8	-82.6	67.1	26.2	-18.7	83.6	117.0	44.9
10971501	133.2	-14.7	11.4	133.2	-3.8	11.4	19.37	12.47	19.22	124.8	-82.6	67.1	26.2	-18.7	83.6	129.1	88.2
10971502	62.0	-44.5	69.5	66.4	-41.8	69.1	19.37	12.47	19.22	124.8	-82.6	67.1	26.2	-18.7	83.6	46.3	73.5
10971503	129.1	-38.0	36.1	133.9	-26.4	37.0	19.37	12.47	19.22	124.8	-82.6	67.1	26.2	-18.7	83.6	115.0	54.5
10971504	76.9	-28.1	57.0	78.9	-25.3	57.4	19.37	12.47	19.22	124.8	-82.6	67.1	26.2	-18.7	83.6	58.0	73.3

10971504 rows × 17 columns

Thus, the descriptive features X, used for training the ANN model were:

- X_0, Y_0, Z_0: initial coordinates of the nodes before loading
- D_x, D_y, D_z: the displacement vectors applied as loading

- $X_{loadnode}$, $Y_{loadnode}$, $Z_{loadnode}$: coordinates of the node on which the displacement vectors were applied
- $X_{BCcentroid}$, $Y_{BCcentroid}$, $Z_{BCcentroid}$: coordinates of the fixed nodes centroid
- Euclidean distance between the node and the centroid of the fixed nodes
- Euclidean distance between the node and the node under loading

lastly, the matrix of the output values, expected to be predicted by the ML model, Y, contained three features and was identified as the coordinates of the nodes after deformation and were introduced to the neural network as X_1, Y_1, and Z_1.

4.4.3 Splitting Strategies

A number of different variations were considered as methods of partitioning the dataset into training and validation subsets. The self-explanatory term of training set refers to the data upon which the machine learning model was trained, whose performance in predicting unseen data could be tested by the validation set. The first method, followed by many authors in the literature [10, 11, 47, 49], was a random selection of the instances in the dataset and their assignment to the training and validation subsets by specific percentages. In this approach, two different shares of the data, 75 $\%$/25 $\%$ corresponding to 8,288,629 training versus 2,742,876 test samples and 85 $\%$/15 $\%$ resulting in 8,777,204 and 2,194,301 samples, were considered for building and evaluation of the models by using the open-source Python library, *Scikit Learn 0.21* [97], which is employed for predictive data analysis.

The first strategy generally led to highly accurate predictions in the work of other authors. However, due to the proximity of a node in the model with its adjacent nodes and similarity in their coordinates, this approach could not be considered an utterly right indication of the neural network capability in replicating the FEM results. Furthermore, there was a high likelihood of sampling a minimum of one simulation related to all of load recipient nodes in the training data, which would question the ability of the neural network model's prediction for a situation in which the load applied to a node, not present in the training data. Thus, another different but novel approach was tested in this thesis for the second splitting strategy. This technique was based on randomly leaving out the entire simulations of a fraction of the 109 nodes undergone loading to test the model's performance when a new node was loaded. Therefore, similar to the first strategy, in this method, 75 $\%$ and 85 $\%$ of the 109 loading nodes equal to the simulations of the respectively 82 and 92 nodes, were set aside for training of the model.

The third approach of building the training and validation sets, aimed at assessing the neural network model's performance in predicting the response of the liver for new loading values. In this method, the simulations of the 109 load nodes were present in both training and validation sets, while only the last three out of 12 (25 $\%$) loading values per load node were set aside for testing the model's performance. Therefore, the neural network's ability to predict the liver's deflections corresponding to a displacement load of 40 mm in three random orientations, applied to all load nodes, could be evaluated.

4.4.4 Feature Scaling

Feature scaling is a technique utilized to normalize the range of independent features of data. In other words, all features in a scaled input are transformed to a specific range. Using this method in a deep neural network for regression increases efficiency and facilitates faster convergence during training. For achieving this purpose, the Scikit-Learn function $MinMaxScaler()$ were applied to the training and validation inputs to uniformly scale them down them between the values of 0 and 1.

4.4.5 Hyperparameter Tuning

The primary and most time-consuming task in obtaining the best result from the neural network was adjusting hyperparameters. Hyperparameters are all the parameters that are defined by the user and are not learned during the training. The hyperparameters identified to have significance in this work, along with their definitions are presented below.

Batch Size

This parameter determines the number of dataset samples propagated through the network before the network's internal parameters, such as weights and biases, are updated. For instance, if the dataset contains 50 samples and the batch size is set to ten, at one time, only ten samples are passed through the network, and this process repeats until the network has seen all the samples. In general, a larger batch size results in shorter training duration, while the demand for a higher computational power and memory increases. On the other hand, choosing a smaller batch size is advantageous since it generally deters the training at stopping at local minima by generating more noise in loss calculation.

Number of Epochs

A cycle of passing forward of the entire training samples in the dataset and their backpropagation through the network is referred to as an epoch. After one epoch is finished, the weights and biases are tuned, which will be used in the next epoch to lower the training loss. This process can continues until no further improvement in the training loss is seen. Therefore, finding the correct number of epochs is of significant importance in both loss reduction and prevention of overtraining. The metric used in the determination of the suitable number of epochs is *Validation Loss*, which as long as it decreases, the training can continue. In case a reduction in the validation loss is not observed after certain epochs, the network training can be terminated by a mechanism known as *Early Stopping*. The optimal region for the number of epochs is shown in Figure 4.14.

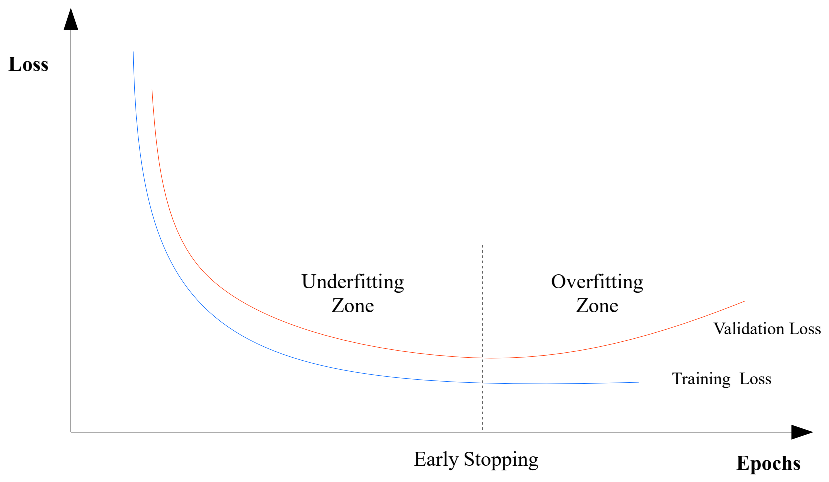


Figure 4.14: The optimal region for choosing the number of epochs

Learning Rate

Learning rate defines the amount by which the weights getting updated in the next epoch. Choosing an appropriate learning rate affects the speed at which the cost function reaches its minimum. An intuitive approach is in the earlier epochs of training, the learning rate has a relatively high value, and as the training progresses, this value decays to prevent overshooting. For achieving this goal, several functions exist that can define a variable learning rate. In this work, however, the function representing the learning rate at each epoch was defined as

$$\Gamma = \frac{\Gamma_0}{1 + decay\ rate \times epoch\ number}, \tag{4.2}$$

where Γ_0 is the initial learning rate, and based on testing different values, the value of 0.001 was chosen. The corresponding plot of this function is also illustrated in Figure 4.15.

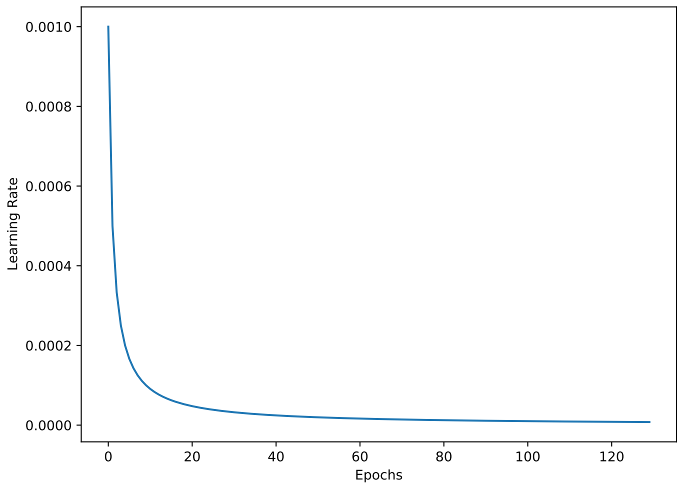


Figure 4.15: Decay of learning rate with epochs

Dropout Rate

A robust neural network algorithm tends to overfit the training data quickly. *Dropout* is a computationally inexpensive technique that helps a deep network to overcome this problem and aids in the improvement of generalization. In the course of training, a dropout rate defines the percentage of a random selection of neurons in a hidden layer to be deactivated, and without being fed to the next layer's neurons, their outputs are ignored. This process pushes the network out of its initial configuration and injects noise into the layer's output. Thus, in a network with the implementation of dropout, the layers have to co-adapt to rectify the mistakes of their previous layers, which can result in the training of a more robust model [98].

In addition to the items introduced above, the following hyperparameters explained in Chapter 2 were also considered for selecting their optimal values/types.

- **Number of hidden layers**
- **Number of neuron per hidden layer**
- **Activation function**
- **Optimizer**
- **Loss function**

For finding the desired hyperparameters, and as the corresponding Python code shown in Listing 4.2 indicates, $GridSearchCV()$, which is a function available in the Scikit-Learn package, was used. This function helps in looping through a provided space of hyperparameters and fits an estimator on a percentage of the training set. The reason for not including the entire training set was based on the significant volume of the data, over which the algorithm was required to test multiple scenarios for a combination of the predefined parameters that demanded high time and computational capacity. For this reason, one percent of the training set was used for finding the best performance of the following hyperparameters out of their predefined values:

- **Epochs**: 30, 60, 120;
- **Batch size**: 32, 64, 128, 256, 512;
- **Neurons**: 64, 128, 256, 512, 1024;
- **Dropout rate**: 0.0, 0.2, 0.4, 0.6;
- **Activation function**: Sigmoid, ReLU, TanH;
- **Optimizer**: SGD, RMSprop, Adam.

```
# Importing required libraries
import tensorflow as tf
import keras
from keras.layers import Dense
from keras.models import Sequential
from tensorflow.keras.layers import Activation
from keras.layers import Dropout
from sklearn.model_selection import GridSearchCV

# Taking a subset (1%) of training set for searching
x_grid, x_not_use, y_grid, y_not_use = train_test_split(xTrain1, yTrain1,
                                                         test_size=0.99,
    random_state=42)
# Finding the shape of input matrix
input_dim=x_grid.shape[1]

# Defining of a function for building a deep learning model
def model_func(activation='relu', optimizer='Adam', dropout_rate=0,
    neurons=0):
    model = Sequential()
    model.add(Dense(neurons, activation=activation , input_dim=input_dim))
    model.add(Dropout(dropout_rate))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(3, activation='linear'))

    # compile the model
    model.compile(optimizer=optimizer, loss='mse', metrics=['acc'])

    return model

# Implement the Scikit_Learn regressor interface that requires model
    defined as a function
from keras.wrappers.scikit_learn import KerasRegressor
model = KerasRegressor(build_fn=model_func, verbose=1)

# Specifying values of hyperparameters by which the performance of the
    model will be tested
```

```
dropout_rate = [0.0, 0.2, 0.4, 0.6]
neurons = [64, 128, 256, 512, 1024]
batch_size = [32, 64, 128, 256, 512]
epochs = [30, 60, 120]
activation = ['sigmoid', 'relu', 'tanh']
optimizer = ['SGD', 'RMSprop', 'Adam']

# Making a dictionary to define the grid space containing all specified
    hyperparamaters
# to explore
param_grid = dict(activation=activation, optimizer=optimizer,
                  dropout_rate=dropout_rate, neurons=neurons,
                  batch_size= batch_size, epochs=epochs)

# n_jobs = 3 uses 3 CPU cores, cross validation = cv
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=3, cv
    =3)

# Every point on the defined grid will be fitted
grid_results = grid.fit(x_grid, y_grid)
```

Listing 4.2: Hyperparameter tuning using GridSearchCV (code adapted from Scikit-Learn documentation [97]

The tuned values of hyperparameters used in the first and second splitting approaches are summarized in Table 4.5.

Table 4.5: Tuned values of hyperparameters for the first and second splitting strategies

Splitting strategy	Number of hidden layer	Number of neurons per hidden layer	Batch size	Dropout	Epochs	Activation function	Optimizer	Loss function
First	2	(256, 256)	32	0.0, 0.0	120	TanH	Adam	MSE
Second	2	(128, 128)	64	0.0, 0.0	420	ReLU	Adam	MAE

It must be noted that, in both strategies, the algorithm returned the highest value of the specified epochs, and technically the learning process could continue. For this reason, in the node-based splitting, this number was manually identified from observing an insignificant change in the loss of the network. Furthermore, the dropout rate was reported back with zero values since overfitting did not occur during training.

The interesting takeaway of this experiment was that the models' best performance was not achieved using the same loss function. While training of the first models using MSE proved efficient, the second model did not respond well to this function and demanded using other alternatives. This lead to the choice of MAE for the training of this model.

Chapter 5

Results and Discussion

The performance of the artificial neural networks designed and configured in this project can be assessed in different ways. However, the most important assessment is to compare the predicted nodal coordinates with those of the finite element method to evaluate the accuracy. The layout illustrated in Figure 5.1 shows the overall logic behind the assessment of the trained ML models.

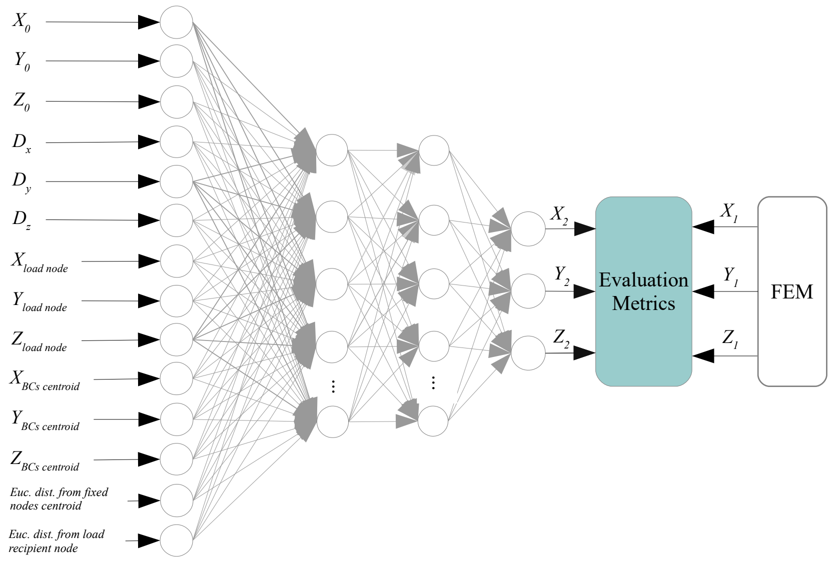


Figure 5.1: The layout of the designed ANNs showing the inputs and comparison of the results with finite element method's nodal coordinates

For the evaluation metrics shown in this figure, several choices similar to the cost functions explained in Chapter 2 existed. Nevertheless, the results in this thesis were generally analyzed using absolute error in directional coordinates and Euclidean error. Another metric, namely *relative error*, for in-depth investigation of the results was also used.

In this chapter, the results of the trained neural networks are presented and analyzed. First, the outcome of each case is presented in the form of graphs and plots, and the key points are discussed and explained. In the last section, the comparative results of the first and second cases of splitting strategies have been put under scrutiny.

5.1 First Splitting Scenario

The first case investigated in the previous chapter utilized a common approach of splitting in machine learning. Namely, randomly sampling out of the instances in the main dataset by a percentage. After adjusting the hyperparameters and running the training for the subcases of 75 and 85 percent for training data, as can be observed from Figure 5.2, the loss of both training and validation data decreased with a close agreement between the curves.

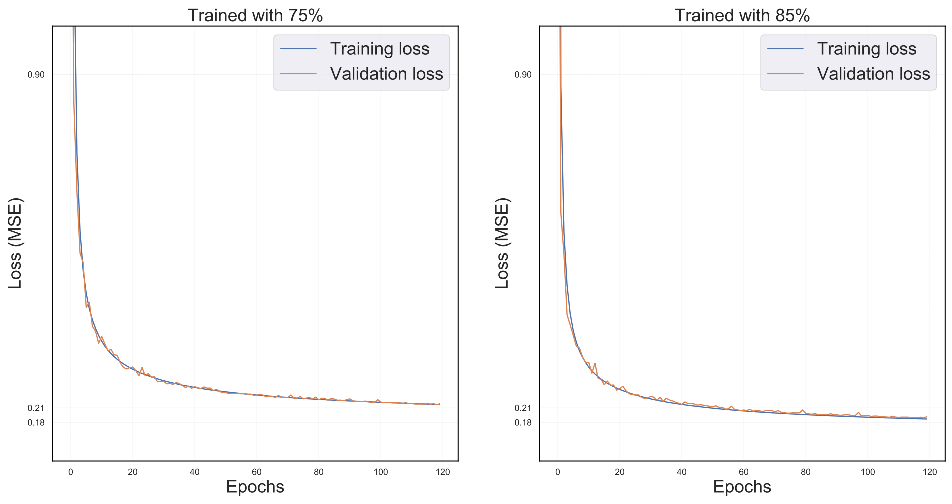


Figure 5.2: Decay of training and validation loss in the first splitting strategy

The 85%/15% case, however, showed a slightly lower mean squared error (MSE) value compared to the 75%/25% splitting strategy by the end of training. A value in the loss function represents the distance between the actual and the predicted location of the nodes. The fitness between the curves also indicates a comparable performance of the model on both training and test data.

The scatter plots in Figure 5.3 and 5.4, show the predicted against the actual coordinates of the nodes in X, Y, and Z directions for two splitting approaches, where the variables with the subscripts of 1 and 2 respectively represent actual and predicted values.

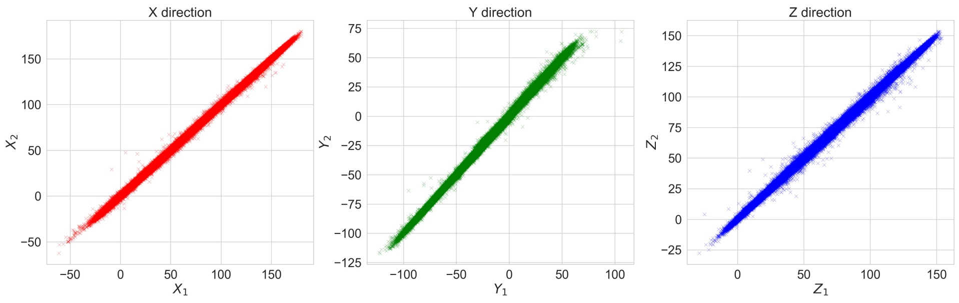


Figure 5.3: Acutal vs. predicted coordinates of the model trained with 75% of the data in the random splitting strategy

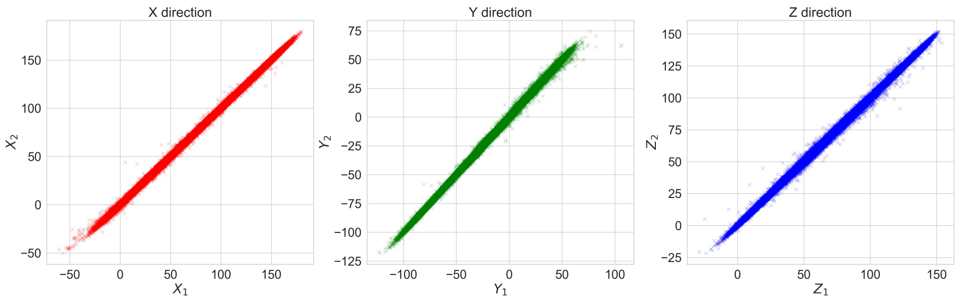


Figure 5.4: Acutal vs. predicted coordinates of the model trained with 85% of the data in the random splitting strategy

As is clear from these visualizations, the majority of the scatters have accumulated to form 45° lines. This orientation indicates that both predicted and actual coordinates have very close values, which is a desirable figure in evaluating ML models. Nevertheless, a minority of the predicted coordinates is far from the actual values. A close examination of the plots also reveals that these far-off predictions are more widespread in the first splitting approach, confirming the outperformance of the model trained with a more significant portion of the dataset.

The distribution of the number of samples with various magnitudes of Euclidean error is presented in Figure 5.5. As is apparent, these illustrations show that the quantity of the nodes with Euclidean errors (E_{Euc}) below 3 mm are predominant within the predictions. It is essential to note that these sub-figures, belonging to training with two dissimilar percentages of data, are not meant to be compared since they are representative of a different number of samples in the validation datasets (1,645,726 equals 15% and 2,742,876, 25% of the samples).

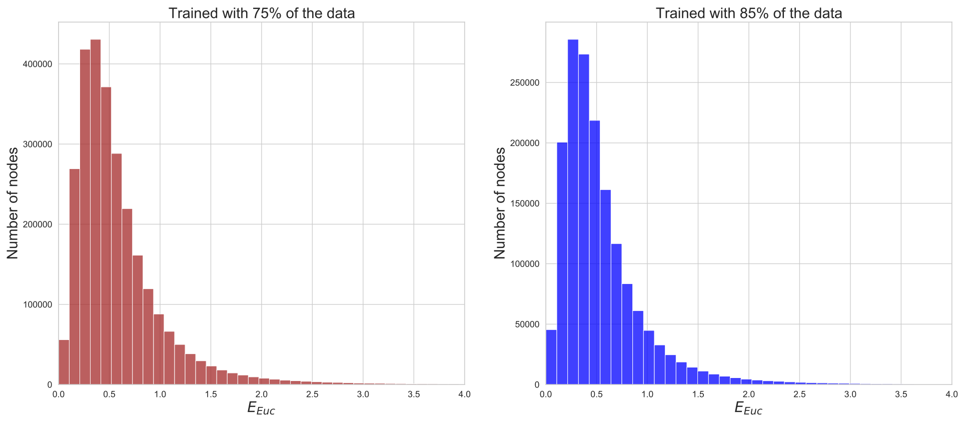


Figure 5.5: Distribution of the samples with respect to the magnitude of their Euclidean errors (mm) in the first splitting strategy

A more comprehensible illustration of the extent and median of directional and Euclidean errors has been depicted in Figure 5.6 and 5.7.

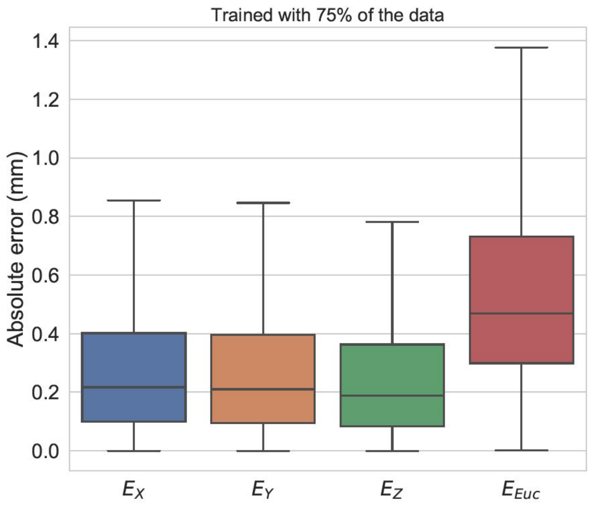


Figure 5.6: Box plots of the absolute errors in the first splitting strategy (sub-case 75%/25%)

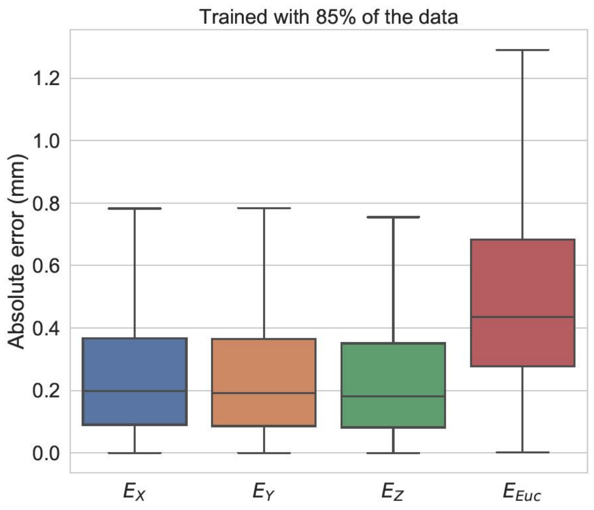


Figure 5.7: Box plots of the absolute errors in the first splitting strategy (sub-case 85%/15%)

These plots indicate the mean absolute error of the samples and divide the error domains into four quartiles. The horizontal lines in the middle of the boxes identify the median of these errors, where the Euclidean errors are lying between 0.4 and 0.5 mm. However, medians of the values corresponding to the 85%/15% case are slightly lower than the case trained with 75% of the dataset. A similar trend can also be seen with regard to the upper extremes of these plots. Nevertheless, a low value around 0.4 mm for the absolute errors does not invariably convey a sense of high accuracy of the machine learning models. One particular reason for this is that a significant share of the nodes can have relatively small displacements, and an error below 1 mm may still be seen as a considerable value. Therefore, a need for a new metric was felt to be implemented to consider the absolute errors and account for the displacement magnitude of a particular node. This metric was decided to be named as *Relative error* and was defined by

$$E_{Relative} = \frac{Euclidean\ error}{displacement\ magnitude}, \tag{5.1}$$

where the relative error shows the percentage of the Euclidean error relative to the magnitude of the displacement magnitude. The box plots in Figure 5.8 show the extent of this new metric for both sub-cases of the first splitting strategy. As is evident from the plots, the medians of the $E_{Relative}$s are close to 15%, indicating that the errors are proportionately smaller than the magnitudes of the displacements. There are some cases, however than the errors are greater than the displacement. Nevertheless, these instances represent a small share of the overall predictions.

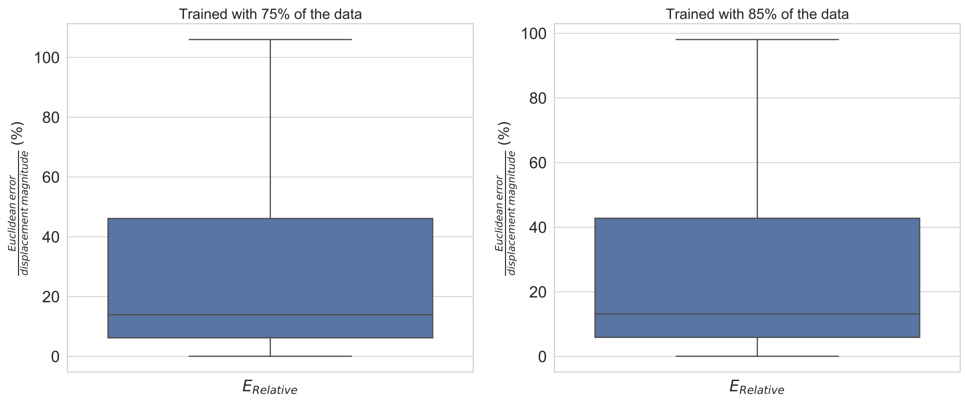


Figure 5.8: Box plots of the relative errors in the first splitting strategy

5.2 Second Splitting Scenario

Training a neural network model using a node-based splitting approach was the greatest challenge of this thesis. Unlike the first approach, the results for the two sub-cases differed significantly, with substantial differences in terms of accuracy. Figure 5.9, shows the drop of the training loss with progress in the epochs number.

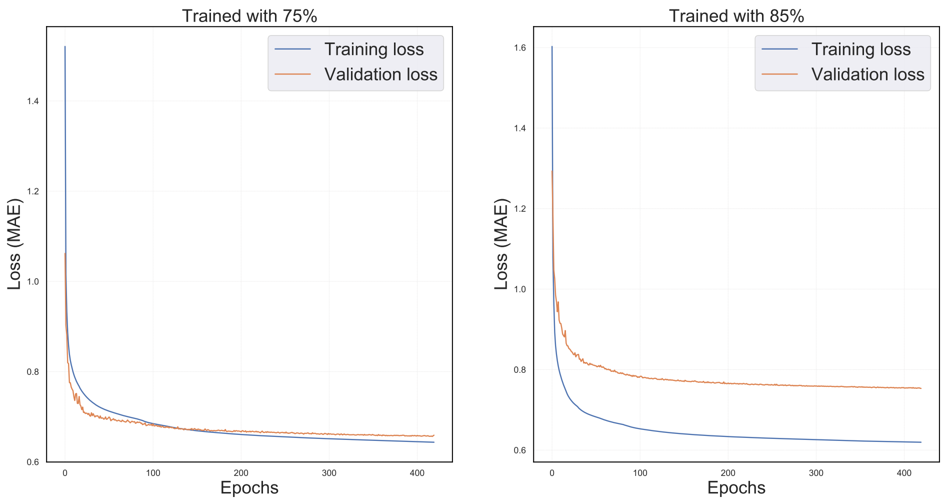


Figure 5.9: Decay of training and validation loss in the node-based splitting strategy

This figures explicitly indicate a better learning process of the model trained with 75% of the dataset, where both training and validation loss closely decreased as the training continued. In contrast, reduction in the validation loss during training for the 85% of the data showed a poorer trend, and the model failed to react decently to the unseen inputs.

The difference between the learning process of both models became more distinct when the actual plot against predicted coordinates was made. Figure 5.10 and 5.11, convey a general overview of the accuracy of these models in prediction.

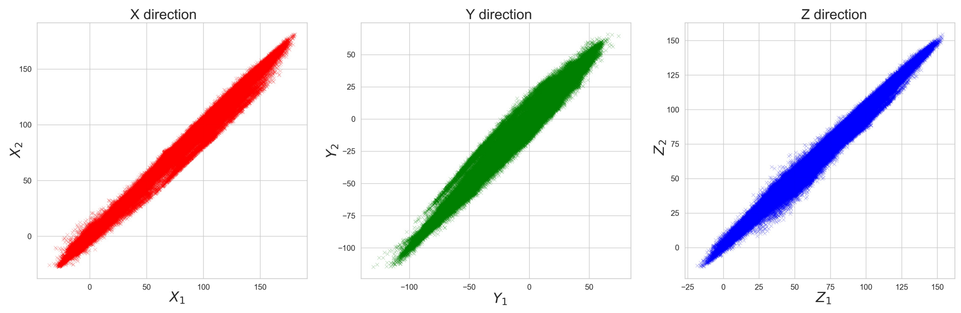


Figure 5.10: Acutal vs. predicted coordinates of the model trained with 75% of the data in the node-based splitting strategy

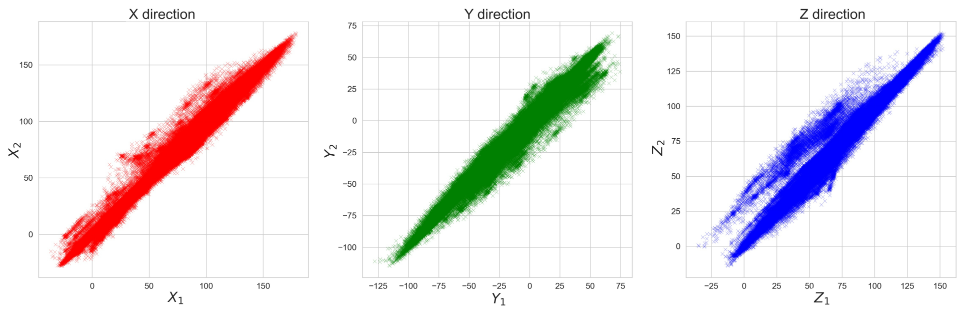


Figure 5.11: Acutal vs. predicted coordinates of the model trained with 85% of the data in the node-based splitting strategy

As expected from the previous plots in Figure 5.9, the first sub-case outperformed the case with a lower percentage of data set aside for validation. This was in contrast to the random splitting approach, where a slightly higher performance was achieved by training with 85% of the data. For this reason, in this subsection, the best model was chosen to proceed with, and sub-case two was excluded in the further analyses and performance check.

Compared to the first splitting approach and as shown in Figure 5.12, the distribution of the Euclidean errors in the second approach covers a more considerable extent of the horizontal axis. This histogram clearly shows that most errors in the prediction of nodal locations remained below 4 mm. At the same time, this range for the first approach was more accurate, with an insignificant number of nodes having Euclidean errors above 2 mm.

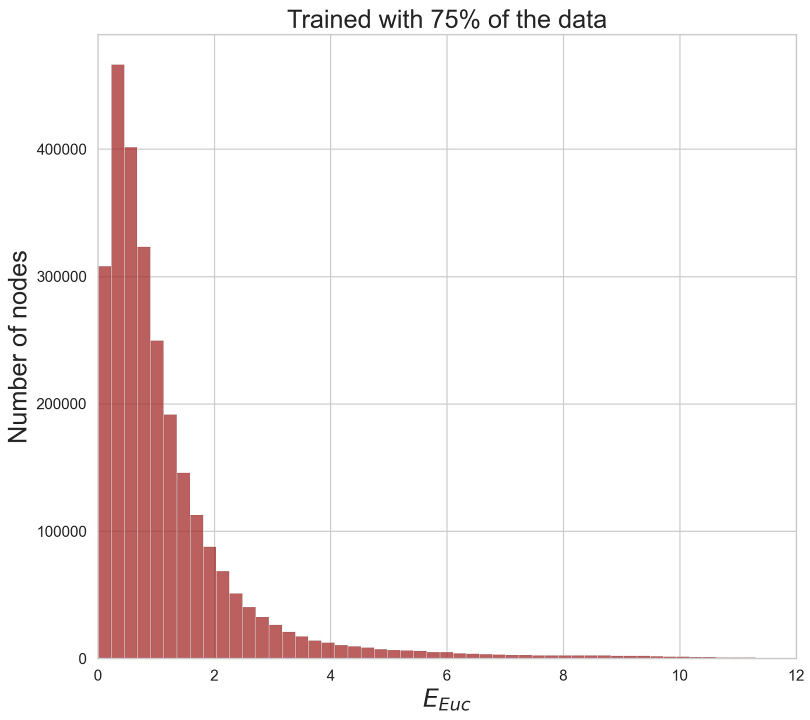


Figure 5.12: Distribution of the samples with respect to the magnitude of their Euclidean errors (mm) in the node-based splitting strategy

The box plots presented in Figure 5.13 provides a more informative aspect of the average size and extent of the directional and Euclidean errors.

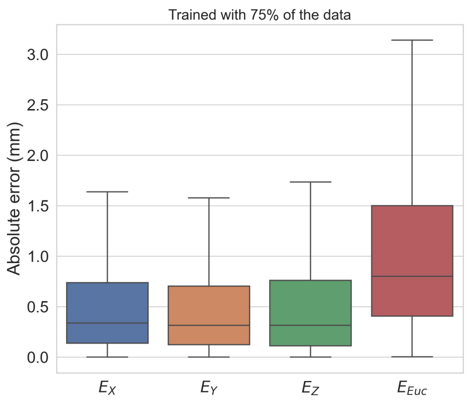


Figure 5.13: Box plots of the absolute errors in the node-based splitting strategy

As the figure shows, the medians of all directional and Euclidean errors were below 1 mm.

Furthermore, the fourth quartile of the Euclidean absolute errors was below ranged from 1.5 mm to slightly above 3 mm, showing that the errors were marginally inside the acceptable threshold for clinical applications [49]. It must be noted that far-off predictions in the form of outliers were also present in the data, which for the sake of better illustration, they were excluded in the box plots.

Finally, in this subsection, relative error pertaining to the model trained with 75% of the data is presented in Figure 5.14. As can be seen, the extent of this metric for this approach is comparatively larger than the first strategy. However, the median of the relative errors is located slightly below 30% as opposed to the first approach with approximately 15%.

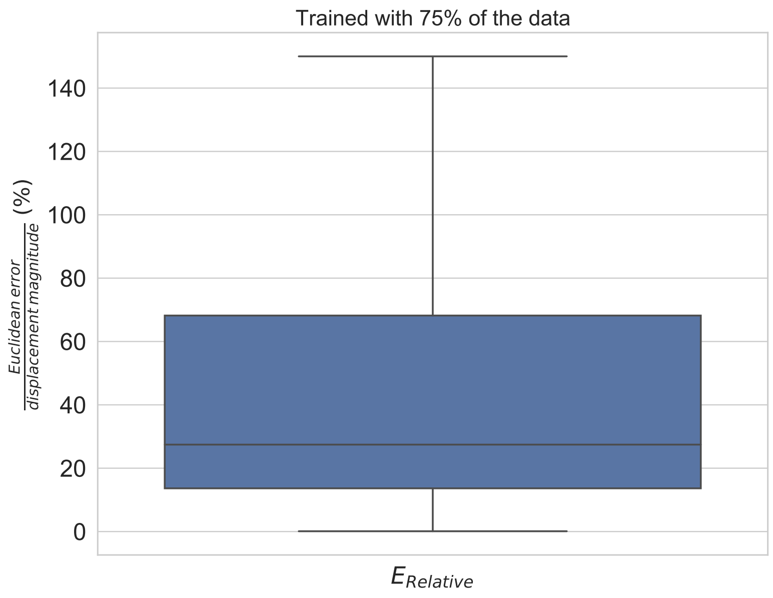


Figure 5.14: Box plots of the relative errors in the node-based splitting strategy

5.3 Comparison of the Results

The summary of all results and noticeable information is presented in Table 5.1.

Table 5.1: Summary of all results: Euclidean Error (EE); Mean Absolute Error (MAE))

Splitting method	MAE (mm)			Percentage of instances			Maximum error (mm)			Training time (sec)	Prediction time (sec)
	X	Y	Z	EE 1 mm	EE 3 mm	EE 5 mm	X	Y	Z		
First (75)	0.308	0.306	0.284	87.184	99.337	99.858	42.47	40.99	30.80	**46137**	0.268
First (85)	**0.283**	**0.282**	**0.273**	**88.945**	**99.477**	**99.884**	38.96	43.89	30.52	51277	0.331
Second (75)	0.675	0.631	0.654	59.367	91.622	96.165	**23.53**	**26.33**	**24.53**	69734	**0.172**

The values specified with bold font are the best performance among the rest of the data. In terms of mean absolute error, the first splitting method trained with 85% of the FEM data outperformed the two other sub-cases. Moreover, this method contained a lower percentage of instances with significant errors. Nonetheless, the node-based method had the smallest maximum error sizes and the prediction time. However, this model required a higher number of epochs for reaching its current accuracy, which resulted in the longest training time among other models.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The primary motivation behind the conduction of studies and experiments in this thesis was to investigate the feasibility of replacing finite element analysis of liver tissues with a trained artificial neural network to proceed towards the requirements necessary in clinical applications. The results presented in this work showed the applicability of this method and backed up the expectations of computational efficiency and accuracy compared with the data generated from FEM. The ANNs model was capable of capturing the deformations caused by displacement loads applied to random nodes on the liver's surface and delivered the outputs in terms of the nodal coordinates, from which the locations of the internal structures, including tumor and the network of vessels, could be determined.

This work also used a novel approach in the training of the ML models using a random node-based splitting approach and presented a comparative analysis of the results with the more conventional random instance-based approach. Although accuracy-wise, the latter showed superior performance, the node-based method demanded shorter prediction time, and the outputs were still within the required threshold for accuracy. Regarding the training time, it is essential to notice that these models were trained on a personal computer with limited computational resources. However, in real-world applications, the utilization of graphics processing units and parallel processing is standard and substantially decreases the training time for both finite element simulations and training of the machine learning models. Thus, even more improvement in the results can be expected.

6.2 Future Work

Due to the time limitation, this thesis predominantly focused on the proof of concept aspect of the project, and some areas can be considered for further research and experiment.

First, due to the geometrical difference of the liver, the inclusion of an automatic algorithm for the segmentation and mesh generation process into the general pipeline seemed to be necessary. This will result in a significant reduction in time and also allows for the training of the machine learning models to be performed with more than one geometry, which in turn increases the generalization capability of the models.

Another significant contribution in future work can be using convolutional layers, known to work well with image processing. This approach is most likely the missing link between capturing the organ images directly from a camera, computing the deformed internal structure, and showing a representative mesh of the organ.

Furthermore, the accuracy of the FEM simulations can be elevated by finding better boundary conditions and accounting for the effects of the neighboring tissues. Providing finding reliable biomechanical test data, the effect of Glisson's capsule may also be included in the FEM simulations.

Bibliography

[1] Kristy K Brock, Laura A Dawson, Michael B Sharpe, Douglas J Moseley, and David A Jaffray. Feasibility of a novel deformable image registration technique to facilitate classification, targeting, and monitoring of tumor and normal tissue. *International Journal of Radiation Oncology* Biology* Physics*, 64(4):1245–1254, 2006.

[2] Rosalie Plantefève, Igor Peterlik, Nazim Haouchine, and Stéphane Cotin. Patient-specific biomechanical modeling for guidance during minimally-invasive hepatic surgery. *Annals of biomedical engineering*, 44(1):139–153, 2016.

[3] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, pages 283–321. Springer, 2012.

[4] Hadrien Courtecuisse, Hoeryong Jung, Jérémie Allard, Christian Duriez, Doo Yong Lee, and Stéphane Cotin. Gpu-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in biophysics and molecular biology*, 103(2-3):159–168, 2010.

[5] Francisco Chinesta, Adrien Leygue, Felipe Bordeu, Jose Vicente Aguado, Elías Cueto, David González, Iciar Alfaro, Amine Ammar, and Antonio Huerta. Pgd-based computational vademecum for efficient design, optimization and control. *Archives of Computational Methods in Engineering*, 20(1):31–59, 2013.

[6] Oscar J Pellicer-Valero, María José Rupérez, Sandra Martínez-Sanchis, and José D Martín-Guerrero. Real-time biomechanical modeling of the liver using machine learning models trained on finite element method simulations. *Expert Systems with Applications*, 143:113083, 2020.

[7] Andrea Mendizabal, Pablo Márquez-Neila, and Stéphane Cotin. Simulation of hyperelastic materials in real-time using deep learning. *Medical image analysis*, 59:101569, 2020.

[8] Tom Gulikers. An integrated machine learning and finite element analysis framework, applied to composite substructures including damage. 2018.

[9] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[10] Michele Tonutti, Gauthier Gras, and Guang-Zhong Yang. A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery. *Artificial intelligence in medicine*, 80:39–47, 2017.

[11] Francisco Martínez-Martínez, María J Rupérez-Moreno, Marcelino Martínez-Sober, Juan Antonio Solves-Llorens, Delia Lorente, AJ Serrano-López, Sandra Martínez-Sanchis, C Monserrat, and José David Martín-Guerrero. A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time. *Computers in biology and medicine*, 90:116–124, 2017.

[12] Stéphanie Marchesseau, Simon Chatelin, and Hervé Delingette. Nonlinear biomechanical model of the liver. In *Biomechanics of living organs*, pages 243–265. Elsevier, 2017.

[13] UCLThe Johns Hopkins University. Liver: Anatomy and functions. `https://www.hopkinsmedicine.org/health/conditions-and-diseases/liver-anatomy-and-functions`.

[14] Veronica Garbar and Bruce W Newton. Anatomy, abdomen and pelvis, falciform ligament. In *StatPearls [Internet]*. StatPearls Publishing, 2019.

[15] Wonhyo Seo and Won-Il Jeong. Hepatic non-parenchymal cells: Master regulators of alcoholic liver disease? *World Journal of Gastroenterology*, 22(4):1348, 2016.

[16] Zbigniew Kmiec. *Cooperation of liver cells in health and disease: with 18 tables*, volume 161. Springer Science & Business Media, 2001.

[17] Pietro Majno, Gilles Mentha, Christian Toso, Philippe Morel, Heinz O Peitgen, and Jean HD Fasel. Anatomy of the liver: an outline with three levels of complexity–a further step towards tailored territorial liver resections. *Journal of hepatology*, 60(3):654–662, 2014.

[18] American Cancer Society. What is bile duct cancer? `https://www.cancer.org/cancer/bile-duct-cancer/about/what-is-bile-duct-cancer.html`.

[19] Nazim Haouchine, Jeremie Dequidt, Igor Peterlik, Erwan Kerrien, Marie-Odile Berger, and Stéphane Cotin. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *2013 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 199–208. IEEE, 2013.

[20] Liver blood supply diagram. `https://www.anatomynote.com/human-anatomy/blood-supplement/liver-blood-supply-diagram`. Accessed: 2021-01-20.

[21] Osamu OHTANI. Three-dimensional organization of the collagen fibrillar framework of the human and rat livers. *Archives of histology and cytology*, 51(5):473–488, 1988.

[22] Esra Roan. The effect of glisson's capsule on the superficial elasticity measurements of the liver. *Journal of biomechanical engineering*, 132(10), 2010.

[23] Patterns of toxic injury. `http://www.toxmsdt.com/topic-6-patho.html`. Accessed: 2021-01-18.

[24] Zhan Gao, Kevin Lister, and Jaydev P Desai. Constitutive modeling of liver tissue: experiment and theory. *Annals of biomedical engineering*, 38(2):505–516, 2010.

[25] Stéphanie Marchesseau, Tobias Heimann, Simon Chatelin, Rémy Willinger, and Hervé Delingette. Fast porous visco-hyperelastic soft tissue model for surgery simulation: application to liver surgery. *Progress in biophysics and molecular biology*, 103(2-3):185–196, 2010.

[26] Cora Wex, Susann Arndt, Anke Stoll, Christiane Bruns, and Yuliya Kupriyanova. Isotropic incompressible hyperelastic models for modelling the mechanical behaviour of biological tissues: a review. *Biomedical Engineering/Biomedizinische Technik*, 60(6):577–592, 2015.

[27] Cheekong Chui, E Kobayashi, X Chen, T Hisada, and I Sakuma. Combined compression and elongation experiments and non-linear modelling of liver tissue for surgical simulation. *Medical and Biological Engineering and Computing*, 42(6):787–798, 2004.

[28] Minh Tuan Duong and Universitätsprofessor Dr-Ing RUS Mikhail Itskov. *Hyperelastic modeling and soft-tissue growth integrated with the smoothed finite element method-SFEM*. PhD thesis, Universitätsbibliothek der RWTH Aachen, 2015.

[29] Sahbi Aloui and Mohammed El Yaagoubi. Determining the compression-equivalent deformation of sbr-based rubber material measured in tensile mode using the finite element method. *Applied Mechanics*, 2(1):195–208, 2021.

[30] Holm Altenbach and Andreas Öchsner. *Encyclopedia of Continuum Mechanics*. Springer Berlin Heidelberg, 2020.

[31] invariant. `https://www.merriam-webster.com/dictionary/invariant`. Accessed: 2021-02-28.

[32] Grégory Chagnon, Marie Rebouah, and Denis Favier. Hyperelastic energy densities for soft biological tissues: a review. *Journal of Elasticity*, 120(2):129–160, 2015.

[33] Michael Smith. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp, United States, 2009.

[34] Gerhard A Holzapfel et al. Biomechanics of soft tissue. *The handbook of materials behavior models*, 3:1049–1063, 2001.

[35] Sagar Umale, Caroline Deck, Nicolas Bourdet, Parag Dhumane, Luc Soler, Jacques Marescaux, and Remy Willinger. Experimental mechanical characterization of abdominal organs: liver, kidney & spleen. *Journal of the mechanical behavior of biomedical materials*, 17:22–33, 2013.

[36] Raymond William Ogden. Large deformation isotropic elasticity–on the correlation of theory and experiment for incompressible rubberlike solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 326(1567):565–584, 1972.

[37] Melvin Mooney. A theory of large elastic deformation. *Journal of applied physics*, 11(9):582–592, 1940.

[38] Oon H Yeoh. Some forms of the strain energy function for rubber. *Rubber Chemistry and technology*, 66(5):754–771, 1993.

[39] Ellen M Arruda and Mary C Boyce. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *Journal of the Mechanics and Physics of Solids*, 41(2):389–412, 1993.

[40] Raymond W Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997.

[41] Shima Zaeimdar. *Mechanical characterization of breast tissue constituents for cancer assessment*. PhD thesis, Applied Sciences: School of Mechatronic Systems Engineering, 2014.

[42] Tie Hu and Jaydev P Desai. Characterization of soft-tissue material properties: large deformation analysis. In *International Symposium on Medical Simulation*, pages 28–37. Springer, 2004.

[43] YB Fu, CK Chui, and CL Teo. Liver tissue characterization from uniaxial stress–strain data using probabilistic and inverse finite element methods. *Journal of The Mechanical Behavior of Biomedical Materials*, 20:105–112, 2013.

[44] Alexandre Hostettler, SA Nicolau, Y Rémond, Jacques Marescaux, and Luc Soler. A real-time predictive simulation of abdominal viscera positions during quiet free breathing. *Progress in biophysics and molecular biology*, 103(2-3):169–184, 2010.

[45] Esra Roan and Kumar Vemaganti. The nonlinear material properties of liver tissue determined from no-slip uniaxial compression experiments. 2007.

[46] HM Yin, LZ Sun, Ge Wang, and Michael W Vannier. Modeling of elastic modulus evolution of cirrhotic human liver. *IEEE Transactions on biomedical engineering*, 51(10):1854–1857, 2004.

[47] José D Martín-Guerrero, María J Rupérez-Moreno, Francisco Martinez-Martínez, Delia Lorente-Garrido, Antonio J Serrano-López, Carlos Monserrat, Sandra Martínez-Sanchis, and Marcelino Martínez-Sober. Machine learning for modeling the biomechanical behavior of human soft tissue. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 247–253. IEEE, 2016.

[48] Costin D Untaroiu and Yuan-Chiao Lu. Material characterization of liver parenchyma using specimen-specific finite element models. *Journal of the mechanical behavior of biomedical materials*, 26:11–22, 2013.

[49] Delia Lorente, Francisco Martínez-Martínez, María José Rupérez, MA Lago, Marcelino Martínez-Sober, Pablo Escandell-Montero, José María Martínez-Martínez, Sandra Martínez-Sanchis, Antonio J Serrano-López, C Monserrat, et al. A framework for modelling the biomechanical behaviour of the human liver during breathing in real time using machine learning. *Expert Systems with Applications*, 71:342–357, 2017.

[50] Kevin Lister, Zhan Gao, and Jaydev P Desai. Development of in vivo constitutive models for liver: Application to surgical simulation. *Annals of biomedical engineering*, 39(3):1060–1073, 2011.

[51] Jung Kim and Mandayam A Srinivasan. Characterization of viscoelastic soft tissue properties from in vivo animal experiments and inverse fe parameter estimation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 599–606. Springer, 2005.

[52] Abbas Samani and Donald Plewes. A method to measure the hyperelastic parameters of ex vivo breast tissue samples. *Physics in Medicine & Biology*, 49(18):4395, 2004.

[53] Badar Rashid, Michel Destrade, and Michael D Gilchrist. Mechanical characterization of brain tissue in simple shear at dynamic strain rates. *Journal of the mechanical behavior of biomedical materials*, 28:71–85, 2013.

[54] Raymond William Ogden. Large deformation isotropic elasticity on the correlation of theory and experiment for incompressible rubberlike solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 326(1567):565–584, 1972.

[55] Amy E Kerdok, Mark P Ottensmeyer, and Robert D Howe. Effects of perfusion on the viscoelastic characteristics of liver. *Journal of Biomechanics*, 39(12):2221–2231, 2006.

[56] Daniel J O'Shea. Hyperelasticity for soft biological tissues and fibre-reinforced composites using orthotropic fourth-order tensors. 2019.

[57] Chen Jiqing, Du Tianya, Lan Fengchong, and Liu Chaoyang. Progress of research on injury biomechanical model and tests of human liver in impact. In *2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 479–484. IEEE, 2016.

[58] Xila Liu and Leiming Zhang. Structural theory. *Bridge Engineering Handbook. Ed. Wai-Fah Chen and Lian Duan Boca Raton: CRC Press*, 2000.

[59] Implicit vs explicit finite element method (fem): What is the difference? `https://www.simscale.com/blog/2019/01/implicit-vs-explicit-fem`. Accessed: 2021-02-30.

[60] Tom M Mitchell et al. Machine learning. 1997.

[61] Machine learning workflow. https://commandstech.com/ml/. Accessed: 2021-03-18.

[62] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[63] What is the difference between classification and regression. https://kindsonthegenius.com/blog/what-is-the-difference-between-classification-and-\regression. Accessed: 2021-01-18.

[64] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.

[65] Andreas C Müller, Sarah Guido, et al. *Introduction to machine learning with Python: a guide for data scientists.* " O'Reilly Media, Inc.", 2016.

[66] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[67] MATLAB. *9.7.0.1190202 (R2019b).* The MathWorks Inc., Natick, Massachusetts, 2018.

[68] Andreas C Neves, Ignacio González, John Leander, and Raid Karoumi. A new approach to damage detection in bridges using machine learning. In *International Conference on Experimental Vibration Analysis for Civil Engineering Structures*, pages 73–84. Springer, 2017.

[69] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[70] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[71] Marvin Minsky and Seymour A Papert. *Perceptrons*. MIT press, Cambridge, MA., 1969.

[72] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. In *Artificial Neural Networks*, pages 14–22. Springer, 2008.

[73] Snehashish Chakraverty and Susmita Mall. *Artificial neural networks for engineers and scientists: solving ordinary differential equations*. CRC Press, 2017.

[74] Sagar Sharma. Activation functions in neural networks. *Towards Data Science*, 6, 2017.

[75] Softmax activation function explained. https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60. Accessed: 2021-01-20.

[76] Cost, activation, loss function—— neural network—— deep learning. what are these? https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-\learning-what-are-these-91167825a4de. Accessed: 2021-03-02.

[77] Chapter 7 training neural networks part 1. https://srdas.github.io/DLBook/GradientDescentTechniques.html. Accessed: 2021-02-10.

[78] Understanding optimizers. https://deeplearningdemystified.com/article/fdl-4. Accessed: 2021-03-19.

[79] Stochastic gradient descent — clearly explained !! https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-\53d239905d31. Accessed: 2021-02-15.

[80] Adam — latest trends in deep learning optimization. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization\-6be9a291375c. Accessed: 2021-02-17.

[81] An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/. Accessed: 2021-02-27.

[82] B Scott Kessler, A Sherif El-Gizawy, and Douglas E Smith. Incorporating neural network material models within finite element analysis for rheological behavior prediction. 2007.

[83] Ken'ichi Morooka, Xian Chen, Ryo Kurazume, Seiichi Uchida, Kenji Hara, Yumi Iwashita, and Makoto Hashizume. Real-time nonlinear fem with neural network for simulating soft organ model deformation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 742–749. Springer, 2008.

[84] Micha Pfeiffer, Carina Riediger, Jürgen Weitz, and Stefanie Speidel. Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks. *International journal of computer assisted radiology and surgery*, 14(7):1147–1155, 2019.

[85] Ct scan vs. mri. https://www.healthline.com/health/ct-scan-vs-mri. Accessed: 2021-03-09.

[86] Peijun Hu, Fa Wu, Jialin Peng, Yuanyuan Bao, Feng Chen, and Dexing Kong. Automatic abdominal multi-organ segmentation using deep convolutional neural network and time-implicit level sets. *International journal of computer assisted radiology and surgery*, 12(3):399–411, 2017.

[87] Hojin Kim, Jinhong Jung, Jieun Kim, Byungchul Cho, Jungwon Kwak, Jeong Yun Jang, Sang-wook Lee, June-Goo Lee, and Sang Min Yoon. Abdominal multi-organ auto-segmentation using 3d-patch-based deep convolutional neural network. *Scientific reports*, 10(1):1–9, 2020.

[88] Fang Lu, Fa Wu, Peijun Hu, Zhiyi Peng, and Dexing Kong. Automatic 3d liver location and segmentation via convolutional neural network and graph cut. *International journal of computer assisted radiology and surgery*, 12(2):171–182, 2017.

[89] 3d slicer image computing platform. `https://www.slicer.org/`. Accessed: 2021-04-07.

[90] Autodesk meshmixer. `https://www.meshmixer.com/`. Accessed: 2021-04-07.

[91] S Umale, C Deck, N Bourdet, M Diana, L Soler, and R Willinger. Modeling and validation of the human liver and kidney models. *The International Research Council on Biomechanics of Injury Conferance*, 2013.

[92] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[93] François Chollet et al. Keras. `https://keras.io`, 2015.

[94] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[95] Michael Waskom and the seaborn development team. mwaskom/seaborn, September 2020.

[96] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[97] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[98] A gentle introduction to dropout for regularizing deep neural networks. `https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks`. Accessed: 2021-03-05.