The application of Neural Networks for classifying impedance spectra established with needle electrode measurements in fat and muscle tissue

by

Andreas Bjurstedt

THESIS

for the degree of

MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences

University of Oslo

Mars 2022

Abstract

Different electrical impedance properties in different types of tissue can be utilized in order to distinguish between several tissue types and to recognize each particular type. Electrical impedance spectra established with a needle electrode, where only the needle tip is uninsulated and active as an electrode, can be used in order to separate between different tissue types. Since the electrode surface area is small, the sensitivity volume of the electrode is also small. The sensitivity has a spherical shape and has the needle tip positioned in its center. The tissue within the sensitivity volume dominates the measured tissue impedance. The small sensitivity volume means that it is the tissue in the immediate vicinity of the needle tip which dominates the measured impedance. The small volume also gives a high probability for the tissue within the volume to be homogenous. For a given current I, an electrode with a small surface area is exposed to a higher current density J than an electrode with a larger surface area. Higher current density entails larger influence from Electrode Polarization Impedance (EPI). The impedance measured with the needle electrode contains EPI in series with the tissue impedance. The influence from EPI is also frequency dependent. Considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz. Instead of just trying to minimize the influence of EPI in the impedance measurements by omitting the frequency range below 50 kHz, the tissue dependencies of the EPI are explored. Two data sets with impedance spectra are established. For each of the two data sets, a separate type of needle electrode is used in order to establish the impedance spectra by performing impedance measurements in many different locations in fat tissue and muscle tissue in a newly deceased pig. The impedance measurements for each set were performed on a separate afternoon/evening at Rikshospitalet in Oslo. 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue in the frequency range between 10 Hz and 1 MHz were established with each of the two needle electrode types. The two data sets were analyzed with two types of Artificial Neu-

ral Networks (ANNs): One Feed-forward Neural Network (FNN) and one Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells. By comparing separate analyzes of the spectra in the lower part of the frequency range between 10 Hz and 2812 Hz with analyzes of the spectra in the higher part of the frequency range between 3556 Hz and 1 MHz, it is possible to see if the EPI has a negative effect on the results in the lower part of the frequency range. The accuracy in the frequency range between 10 Hz and 2812 Hz are as good as, or almost as good as the accuracy in the frequency range between 3556 Hz and 1 MHz. The conclusion is then that if the influence from EPI is considerable in the impedance measurements for frequencies up to 10 kHz – 50 kHz, the EPI in the frequency range between 10 Hz and 2812 Hz must be tissue dependent.

Acknowledgements

I would like to thank my supervisor Prof. Ørjan Grøttem Martinsen and my co-supervisor PhD Håvard Kalvøy for an interesting and instructive master project. Thank you for the guidance and support during my work with the project. I would also like to thank Håvard for his help and guidance during the two afternoons/evenings with impedance measurements at Rikshospitalet in Oslo.

I would like to thank PhD Sisay Mebre Abie for spending time, during the last period of the work with his PhD, on showing me how to perform impedance measurements with the Sciospec ISX-3 impedance analyzer and the MFIA impedance analyzer from Zurich Instruments.

Learning what I needed to know about Artificial Neural Networks (ANNs) was the main challenge for me during the work with my master project, because I had no courses in, or experience with, Machine Learning prior to the start of the project. I would like to thank PhD student Jie Hou for showing me examples on how to use the Keras API in order to implement the ANNs. During a period prior to her guidance, I was really struggling with the technicalities in Tensorflow 2, but with the Keras API in addition, the ANNs became much easier to implement.

Contents

Chapter 1

Introduction

Different electrical impedance properties in different types of tissue can be utilized in order to distinguish between several tissue types and to recognize each particular type. As an example: With electrodes at both hands and both feets, these tissue dependent impedance properties can be used to find the body composition in terms of bone, fat, muscle and water. Such a body composition gives a much more useful assessment of the overall health of a person than the weight and the calculation of the person's Body Mass Index (BMI). InBody is an example of a company which exploit tissue dependent impedance properties in order to assess the body composition. It is not uncommon for larger fitness centers to offer assessment of the body composition with equipment which analyses electrical impedance measurements from the body, so it is more available than finding the body composition through for example a Dual-Energy X-ray Absorptiometry (DEXA or DXA) scan. [1] and [2] are two examples of articles which compares results of body composition assessments from Bio-electrical Impedance Analyses (BIA) with DEXA scans as a reference standard. Another example is a method described in [3], where electrical impedance measurements are used in order to predict contact between the tip of the catheter electrode and the endocardium tissue inside the heart in conjunction with RF cardiac ablation. RF cardiac ablation is a surgical treatment against frequently recurring episodes of atrial fibrillation. According to [4], close monitoring of the tissue impedance during the RF ablation may also be used as a marker for durable ablation success, since an impedance decrease of 10 Ohm (Ω) or more indicates that local heating of the tissue in the close vicinity of the catheter electrode has occurred.

The foundation of this thesis is the work performed by Kalvøy et al in [5].

In this article, two needle electrode types, where only the tip is uninsulated and active as an electrode, were used in order to determine their ability to distinguish between different tissue types through electrical impedance measurements. The impedance measurements were performed on a pig under anesthesia, so the measurements were *in vivo*. The Solartron 1260/1297 system was used in order to perform the impedance measurements. The measurements consisted of impedance spectra with 20 measurement points on a logarithmic scale, with four points in each of the 5 decades between 10 Hz and 1 MHz, and with a voltage of 30 mV rms (the root mean square of the peak voltage value). First the two needle electrode types measured the impedance in 4 to 6 positions in muscle tissue and in fat tissue respectively. The plots of the impedance modulus $|Z|$ and phase angle θ showed different patterns for the two tissue types at high frequencies above 100 kHz, which indicates that it is possible to distinguish muscle tissue and fat tissue with the two needle electrode types, at least at the higher part of the frequency range. Then a partial least-squares discriminate analysis and a regression analysis were performed, by applying the Unscrambler 9.7 software from Camo Inc., on up to 12 impedance spectra in muscle tissue, fat tissue and subdermal fat tissue respectively. The analyzes showed that the impedance spectra in muscle tissue can be separated from the impedance spectra in fat and subdermal fat tissue.

A needle electrode at the uninsulated tip of a needle, or the tip of other types of clinical equipment for invasive use, which can provide impedance measurements in order to determine the type of tissue in the immediate vicinity of the tip of the needle/invasive equipment, can be useful in many clinical applications. Examples mentioned in the article ([5]) are biopsies, ablation in small volumes (RF cardiac ablation is already mentioned) or accurate positioning of a needle for drug delivery in the correct type of tissue (a blood vessel, a small volume of muscle or fat tissue or a nerve). The article also indicates that methods based on impedance measurements may be able to measure tissue differences which can't be detected with more expensive, image-based technologies.

The article ([5]) explains that the tissue volume in the immediate vicinity of the uninsulated needle tip, which is the electrode, dominates the measured tissue impedance. The volume, which has a spherical shape and has the needle tip positioned in its center, is called the *sensitivity volume* or the *sensitivity zone*. The measured tissue impedance is some sort of an average of the actual impedance of the tissues in the sensitivity volume. In order to

determine the tissue type in the immediate vicinity of the needle tip with the information from the impedance measurements, the radius of the sensitivity volume needs to be small. This is because a small sensitivity volume gives a high probability for the tissue within the volume to be homogeneous. In conjunction with the radius of the sensitivity volume, the article ([5]) discusses the electrode surface area of the needle and Electrode Polarization Impedance (EPI). The need for the needle electrode to be unipolar in order to achieve proper impedance measurements, and how to ensure it is unipolar, are also discussed in the article. Chapter 2 in this thesis covers how to ensure unipolarity of the needle electrode during the impedance measurements.

A small electrode surface area gives a small sensitivity volume. This can be explained with an analytical example with an electrode which has the shape of a hemisphere or a sphere, see [6]. A hemisphere can resemble the uninsulated tip of a needle. Chapter 2 in this thesis covers such an analytical example. An experiment performed in [5] shows that the radius of the sensitivity volume is small for a needle electrode in saline. In the experiment, an electrode with a large surface area relative to the tip of the needle, was placed in the bottom of a tank with saline (0.9 percent NaCl). The frequency in the experiment was 100 kHz with a voltage of 30 mV rms. The needle was solid with a conical tip and an electrode area of 0.3 mm^2. The result from the experiment suggests that about 97 percent of the impedance modulus $|z|$ is covered by a spherical sensitivity volume with the center at the needle tip and with a radius of 2.6 mm. The conclusion [5] draws is that the major part of $|z|$ is within a spherical sensitivity volume with a radius between 2 mm and 3 mm for the needle type used in the experiment. A Finite Element Model in Comsol Multiphysics of the saline and the needle with a thin insulation layer at the needle shaft and a thin EPI layer at the uninsulated needle tip, was established and analysed in [7]. The result of the Finite Element Analyis is a spherical sensitivity volume with a radius of 3.75 mm from the tip of the needle, which covers 97 percent of $|z|$ in the saline. The difference between the Finite Element Analysis and the experiment is considered to be due to necessary simplifications of the needle geometry, with the thin insulation and EPI layer, in the Finite Element Model.

When current I flows through two Current Carrying (CC) electrodes and the electrolyte solution between them, the conversion between conduction with electrones and conduction with ions occurs at the interface between the CC electrode surfaces and the electrolyte solution. A side effect of the conversion between electrones and ions as current carriers is polarization of

the two electrodes, which means Electrode Polarization Impedance (EPI) and change in voltage at the two electrode surfaces, see [6]. The impedance measured with the needle electrode therefore contains EPI in series with the tissue impedance. The current density field **J** is directed radially towards the surface or away from the surface of an electrode, and the size of the field is the current density $J = I/A$ where A is the surface area of the electrode. For a given current I, an electrode with a small surface area is therefore exposed to a higher current density than an electrode with a larger surface area. It is similar for the EPI. [6] states that the high current density at the surface of a small electrode entails high EPI and a large change in voltage at the surface. For a spherical electrode for example, the ratio between the EPI and the tissue impedance is inversely proportional to the electrode radius, according to the book ([6]). Since the surface area of an uninsulated needle tip is small, in order to achieve a sensitivity volume with a small radius, the contribution from EPI in tissue impedance measurements can't be neglected.

The influence from EPI is frequency dependent. The electro chemical background explaining the EPI isn't a topic in this thesis. Still it is not unreasonable to accept that the effects which causes the EPI at the electrode surface have more time to establish themselves at low frequencies than at high frequencies. [5] refers to several articles concerning EPI influence on impedance measurements, and from these sources the conclusion drawn is that considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz. Further, the article refers to the work in [8] where EPI is found to be dependent on the cell concentration in a suspension. Therefore a method for estimation of EPI in impedance measurements has to be tailored specifically for all types of tissue or cell suspension consentrations. [5] suggest that trying to exploit tissue dependencies in the EPI in order to distinguish between different types of tissue is a better approach.

1.1 The purpose and aim of this thesis

As already described, the frequency range in the impedance spectra in the article ([5]) is between 10 Hz and 1 MHz. With 5 decades and 4 measurement points in each decade, more than half of the 20 measuring points were under considerable influence from EPI during the measurements. Still, both a partial least-squares discriminate analysis and a regression analysis with

the Unscrambler 9.7 software from Camo Inc. were able to separate the impedance spectra in muscle tissue from the impedance spectra in fat and subdermal fat tissue.

In this thesis the tissue dependencies in the EPI are explored further. Two data sets with impedance spectra are established. For each of the two data sets, a separate type of needle electrode is used in order to establish the impedance spectra by performing impedance measurements in many different locations in fat tissue and muscle tissue in a newly deceased pig. The impedance measurements for each set were performed on a separate afternoon/evening at Rikshospitalet in Oslo after other researches had completed their research work on the pig when it was under anesthesia, and the pig was put to death. 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue in the frequency range between 10 Hz and 1 MHz were established with each of the two needle electrode types. Peak voltage value was 50 mV, which is 35.4 mV rms. Each impedance spectrum contains 50 measurement points on a logarithmic scale, with 10 points in each of the 5 decades between 10 Hz and 1 MHz. The two data sets with impedance spectra are analysed with a type of Machine Learning called Artificial Neural Networks (ANNs). Both a Feed Feed-forward Neural Network (FNN) and a Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM) cells are used in the analyzes. Unlike the FNN, the RNN with LSTM cells is able to take advantage of the time dependency between the measurement points in the spectra, if there is any, in order to improve the accuracy in the analyzes. On the other hand, the FNN is a more natural starting point, since it is easier to understand. An analysis with a FNN is also considerably faster compared to an analysis with a RNN. The ANNs are described in Chapter 4. Three analyzes are performed with the two types of ANN for the impedance spectra in both data sets: One analysis in the entire frequency range between 10 Hz and 1 MHz, one analysis in the lower part of the frequency range between 10 Hz and 2812 Hz and one analysis in the higher part of the frequency range between 3556 Hz and 1 MHz. Since considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz, it is not unnatural to assume considerable influence from EPI on the impedance measurements in the frequency range between 10 Hz and 2812 Hz. Both the frequency range between 10 Hz and 2812 Hz and the frequency range between 3556 Hz and 1 MHz contain 25 of the 50 impedance measurement points. Then the same number of measurement points are available for the ANNs from the impedance spectra in these two frequency ranges, and therefore the results from these two frequency ranges

can be compared. By comparing the accuracy of the analyzes in the lower part of the frequency range between 10 Hz and 2812 Hz with the accuracy of the analyzes in the higher part of the frequency range between 3556 Hz and 1 MHz, it is possible to make an assessment of the influence from EPI on the analyzes in the frequency range between 10 Hz and 2812 Hz. If the results are as good, or almost as good, for the part of the impedance spectra between 10 Hz and 2812 Hz compared to the part of the impedance spectra between 3556 Hz and 1 MHz, the results suggest that the EPI is dependent of the tissue type.

Prior to the impedance measurements in the fat and muscle tissue in the deceased pig was performed and the two data sets with impedance spectra were established, three needle types and two impedance analyzers were tested. The tests were performed in saline, in two pork chops and in two pieces of side meat. These tests are described in chapter 3. Chapter 2, which is already mentioned, covers how to ensure unipolarity of the needle electrode during impedance measurements. Chapter 4, which covers the Artificial Neural Networks (ANNs), is also mentioned already. Chapter 5 covers the work with the impedance measurements at Rikshospitalet and the organization of the two data sets with the impedance spectra. Chapter 6 covers the results.

Chapter 2

Impedance, unipolarity and the sensitivity volume

2.1 Introduction

This chapter contains three sections with theoretical background related to the electrical impedance measurements with a needle electrode. Section 2.2 gives a basic explanation of the impedance which is measured. Section 2.3 covers how to ensure unipolarity of the needle electrode during the impedance measurements. Although only a setup with three electrodes are used in the impedance measurements in this thesis, the section also includes how unipolarity is ensured in a setup with two electrodes. Section 2.4 covers analytical expressions for the sensitivity volume around an unipolar electrode with the shape of a hemisphere or a sphere.

2.2 The measured impedance

The definition of impedance and reading material about impedance considering electrical circuits can be found in textbooks about electromagnetism, electrical/electronic components and electrical circuits. An example of a textbook with coverage of impedance in circuits is [9].

In a electrical circuit exposed to Alternating Current (AC), resistors, capacitors and inductors are opposing the current flow in the circuit. This current opposition is called *impedance* Z. The current opposition from a resistor is the *resistance* R, which is independent of the frequency f of the voltage source and the alternating current. The current opposition from a

capacitor is the *capacitive reactance*

$$X_C = \frac{-j}{2\pi fC} = \frac{-j}{\omega C}$$

where j is the imaginary number which satisfies $j^2 = -1$, ω is the angular frequency and C is the capacitance of the capacitator. The current opposition from an inductor is the *inductive reactance*

$$X_L = j2\pi fL = j\omega L$$

where L is the inductance of the inductor. The unit of R, X_C and X_L is Ω (Ohm). So the impedance of an electrical circuit is dependent of the frequency f of the voltage source and the alternating current, and is expressed in the complex plane as a complex number with a real part and an imaginary part. The polar form of the impedance also includes the phase angle θ, which states the phase difference between the current in the circuit and the voltage source.

The needle measures the impedance in the tissue in the immediate vicinity of the uninsulated tip of the needle, which is the electrode. The measured impedance also contains Electrode Polarization Impedance (EPI) in series with the tissue impedance, see [6] and [5]. In the same way as for an electrical cicruit, the measured impedance is dependent of the frequency f of the voltage source and the alternating current. As an example, [6] explains that a trait of cell membranes is high capacitance, which causes low frequency current to flow around the cells, while current with higher frequency can pass through the membranes due to lower capacitive reactance at higher frequencies.

The impedance is expressed in the complex plane as a complex number with a real part and an imaginary part

$$Z = Z_x + jZ_y \tag{2.1}$$

The polar form of the impedance is described by the modulus

$$|Z| = \sqrt{Z_x^2 + Z_y^2} \tag{2.2}$$

and the phase angle, which can be expressed in the following three ways

$$\theta = cos^{-1}\left(\frac{Z_x}{|Z|}\right) \quad \text{or}$$
$$\theta = sin^{-1}\left(\frac{Z_y}{|Z|}\right) \quad \text{or}$$
$$\theta = tan^{-1}\left(\frac{Z_y}{Z_x}\right) \tag{2.3}$$

Figure below shows an illustration of the impedance in the complex plane

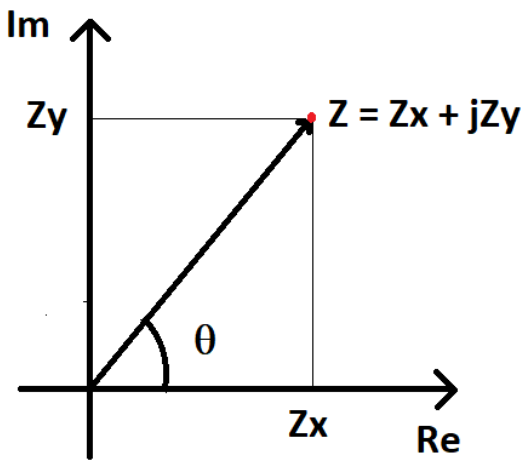

Figure 2.1: The impedance $Z = Z_x + jZ_y$ in the complex plane. *Re* means the real x-axis and *Im* means the imaginary y-axis.

2.3 Unipolarity of the needle electrodes

The conductivity of the body is due to electrolyte solutions, which is liquids containing cations and anions, see [6]. Important cations in the body are for example Sodium Na^+ and Potassium K^+ and important anions are for example Chloride Cl^- and Bicarbonate HCO_3^-. Electrolyte solutions are inside every cell and around every cell in the body. When an electrolyte solution is a part of an electrical circuit, the transition between conduction with electrones and conduction with ions occurs at the interface between the electrode surface and the electrolyte solution. In order to form a closed

circuit where current can flow, two Current Carrying (CC) electrodes are required. The voltage (or potential) between two positions in the electrolyte solution is measured between two electrodes called Pick Up (PU) electrodes, and it is measured at the PU electrode called *Working Sense* with respect to the other PU electrode called *Reference* or *Neutral*. When the current is known also, the impedance between the Working Sense electrode and the Reference electrode can be determined. Here is three different electrode setups with the two CC electrodes and the two PU electrodes:

- A setup with two electrodes where the electrodes are both CC electrodes and PU electrodes.
- A setup with three electrodes where the Working Sense PU electrode also is a CC electrode. The other CC electrode and the Reference PU electrode are two separate electrodes.
- A setup with four electrodes, consisting of two CC electrodes and two PU electrodes.

In this thesis only the setup with three electrodes will be used. In [5] the setup with two electrodes was used in an experiment in a tank filled with saline (0.9 percent NaCl). The experiment is briefly described in Chapter 1 in this thesis. The experiment shows that the radius of the sensitivity volume is small for a needle electrode in saline. Therefore the setup with two electrodes will be discussed briefly in addition to the setup with three electrodes.

A side effect of the conversion between electrones and ions as current carriers at the interface between the CC electrode surfaces and the electrolyte solution is polarization of the two electrodes, which means EPI and a change in voltage at the two electrode surfaces, see [6]. In a setup with only two electrodes, the electrodes are both CC electrodes and PU electrodes. The measured voltage therefore contains the voltage change at both electrode surfaces together with the voltage change in the electrolyte solution between the two electrodes. The result is impedance measurements which contain the EPI from both electrodes together with the impedance in the electrolyte solution between the two electrodes. In the experiment with two electrodes in a tank filled with saline in [5], the Working Sense electrode was a needle electrode. The other electrode (the Reference electrode) had a considerable larger surface area than the needle electrode. In a electrode setup like this, the current density J at the surface of the large electrode is negligible

compared to J at the surface of the needle electrode. The small current density at the surface of the large electrode also means that the EPI and the voltage change are negligible here compared to EPI and voltage change at the surface of the needle electrode, see [6]. The dominating current density and voltage change at the surface of the needle electrode, entails that this electrode dominates the impedance measurements in terms of volume sensitivity and EPI. Due to this domination, the needle electrode is called *unipolar*.

In [5] a setup with three electrodes was used for the impedance measurements in muscle and fat tissue in a pig under anesthesia. The setup, which ensures unipolarity of the needle electrode, is explained both in the article and in [6]. In this thesis, only the setup with three electrodes is used. In the setup the needle electrode is a CC electrode and the Working Sense PU electrode, just as in the setup with two electrodes. The other CC electrode and the Reference PU electrode are two separate electrodes designed for Electrocardiography (ECG) measurements. The type is: Ambu BlueSensor Q Ag/AgCl ECG electrode. These electrodes are designed to stick to the human skin and are therefore a good choice for skin electrodes during the impedance measurements in muscle and fat tissue in a newly deceased pig too. This setup with three electrodes is also used in tests with saline and in tests with samples of meat. These tests are described in chapter 3

In order to perform the impedance measurements, two impedance analyzers are tested, see chapter 3. The block diagram in figure (2.2) with a signal source and an operational amplifier, is an illustration of how the setup with three electrodes works together with the impedance analyzer. This block diagram can also be found in [5] and in [6].

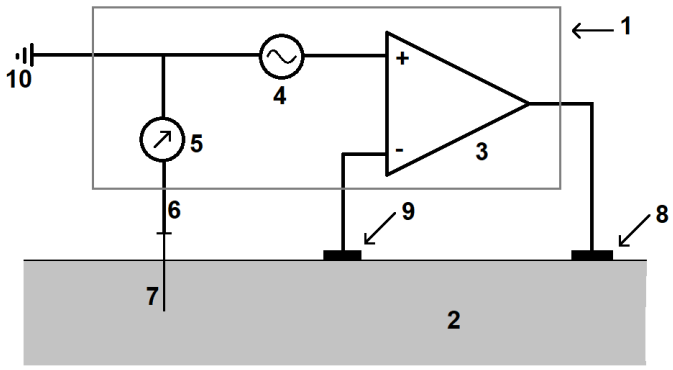

Figure 2.2: An illustration of a setup with three electrodes together with an impedance analyzer. For an explanation of the different symbols, see below

In figure (2.2), the symbols illustrates:

1. The grey frame contains a simple illustration of the impedance analyzer.
2. The grey area illustrates the skin and underlying tissue of a pig.
3. Operational amplifier.
4. Voltage signal source.
5. Measurement of the current in the electrode lead for the needle electrode.
6. Electrode lead for the needle electrode.
7. Needle electrode.
8. Current Carrying (CC) electrode.
9. Reference Pick Up (PU) electrode.
10. Voltage reference point: Earth ground.

The impedance analyzer has four ports which are connected to the electrodes. The needle electrode (7) is connected to the Working Sense port for

the measurement of the voltage between the needle electrode and the Reference PU electrode (9), and to the Current Carrying port called *Working* in order for the needle electrode to be a CC electrode. Figure (2.2) shows that the needle is connected to ground (10), so the potential is $0V$ in the lead (6) for the needle electrode. The other CC electrode is (8) and it is connected to the port called *Counter*. The Reference PU electrode (9) is connected to the Reference port.

The current in the circuit is measured in the lead (6) for the needle electrode (7). Since the Counter CC electrode (8) and the Reference PU electrode (9) are two separate electrodes, the EPI at the Counter CC electrode doesn't have any influence on the voltage measurement between the needle electrode and the Reference PU electrode. The operational amplifier (3) ensures that no current is passing trough the Reference PU electrode. Therefore there is not any EPI and voltage change at the surface of this electrode. The only EPI and voltage change which affects the impedance measurements is at the surface of the needle electrode. The operational amplifier also ensures that the voltage at the Reference PU electrode is the same as the signal source voltage (4) at any time. With no influence on the impedance measurements from the Counter CC electrode and the Reference PU electrode, the needle electrode is unipolar.

2.4 Analytical expressions for the sensitivity volume

In chapter 1, the spherical sensitivity volume around the uninsulated tip of the needle, which is the electrode, is introduced, and it is explained why the radius of the sensitivity volume has to be small in order to be able to determine the tissue type in the immediate vicinity of the needle tip. It is because the tissue within the sensitivity volume dominates the measured impedance. A small electrode surface area gives a sensitivity volume with a small radius. In chapter 1, the experiment in [5] with a needle electrode and a large Reference electrode in a tank filled with saline (0.9 percent NaCl) is briefly described. The experiment suggest that 97 percent of the impedance modulus is covered by a sensitivity volume with a radius of 2.6 mm. Chapter 1 also refers to the Finite Element Analysis in [7], which confirms that the sensitivity volume around the needle tip is of this magnitude.

Analytical expressions for electrodes with the shape of a hemisphere and a

sphere can be used as an example in order to explain that a small electrode surface area gives a sensitivity volume with a small radius, see [6]. Due to the assumptions included in such analytical examples in order to simplify them, they can only be regarded as estimates compared with more accurate Finite Element Analyzes and experiments. The assumptions included are:

- The geometry of the uninsulated tip of the needle is simplified into a hemisphere or a sphere.
- The potential or voltage on the surface of the electrode is known. The other electrical properties of the electrode is omitted.
- EPI is omitted.
- The electrode is unipolar. Any possible influence from the other CC electrode and Reference PU electrode, which in infinitely far away, is omitted.
- Direct-Current (DC) and resistance replaces Alternating Current (AC) and impedance.
- The electrode is submerged into an electrolyte solution which extends infinitely in all directions from the electrode surface.

Assume that the electrode surface has the shape of a hemisphere. First only the uninsulated tip of the needle is submerged into the electrolyte solution. The resistance of the electrolyte solution within a hemispherical shell on the outside of the electrode surface then is:

$$R = \frac{\rho}{2\pi}\left(\frac{1}{a} - \frac{1}{r}\right) \tag{2.4}$$

where a is the radius of the electrode, r is the radius of the hemispherical shell on the outside of the electrode and ρ is the resistivity of the electrolyte solution. The unit of ρ is Ωm. Then the needle electrode is submerged infinitely deep into the electrolyte solution. The electrode surface may now be looked upon as a sphere instead of a hemisphere. The resistance of the electrolyte solution within a spherical shell on the outside of the electrode surface then is:

$$R = \frac{\rho}{4\pi}\left(\frac{1}{a} - \frac{1}{r}\right) \tag{2.5}$$

From equation (2.5), the resistance of the electrolyte solution within the spherical shell when its radius $r \to \infty$ is

$$\lim_{x\to\infty} R = \frac{\rho}{4\pi a} \tag{2.6}$$

The resistance in the electrolyte solution within the spherical shell is

$$R = \frac{\rho}{4\pi}\left(\frac{1}{a} - \frac{1}{10a}\right) = \frac{9}{10}\frac{\rho}{4\pi a} \quad \text{when} \quad r = 10a \tag{2.7}$$

which is 90 percent of the total resistance in the electrolyte solution given in equation (2.6). This means that the electrolyte solution within the sensitivity volume with radius $r = 10a$ dominates with 90 percent of the total resistance in the electrolyte solution. Equation (2.7) is an analytical example which shows that the radius a, and hence the surface area of the electrode, is decisive for the radius of the sensitivity volume. The equation gives an example of that a small electrode surface area gives a sensitivity volume with a small radius.

By introducing a conductive plate at a distance $l = L/2$ from the center of the spherical shaped electrode, the example looks a bit more like the setup of the experiment in [5] with a needle electrode and a large Reference electrode in a tank filled with saline (0.9 percent NaCl). $L = 2l$ is used in equation (2.8) below. Figure (2.3) shows the spherical shaped electrode with radius a and the conductive plate in the electrolyte solution, which now is saline. The plate extends infinitely in all horizontal directions. The saline extends infinitely in all directions above the plate. The figure also shows a spherical shell with radius $l = L/2$ on the outside of the electrode surface.

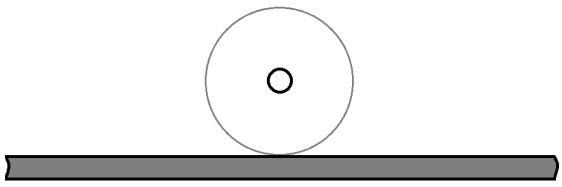

Figure 2.3: The spherical shaped electrode with radius a, the sperical shell with radius $l = L/2$ on the outside of the electrode surface and the conductive plate at a distance $l = L/2$ from the center of the spherical shaped electrode. Saline extends infinitely in all directions above the plate.

In [6], the conductance of the electrolyte solution within the spherical shell on the outside of the electrode surface in figure (2.3) is given as

$$G = 4\sigma\pi a \left[1 + \frac{a}{L} + \left(\frac{a}{L}\right)^2 + \sum_{n=1}^{\infty} n \left(\frac{a}{L}\right)^{n+2} \right] \quad \text{where} \quad L > 2a \tag{2.8}$$

σ is the conductivity of the electrolyte solution, which is saline (0.9 percent NaCl) here. $\sigma = 1.3\, S/m$, which is the same value that was used in the Finite Element Model in [7] of the experiment in [5] with a needle electrode and a large Reference electrode in a tank filled with saline. According to [10] the conductivity of saline varies with temperature. $\sigma = 1.3\, S/m$ is within the range of the conductivity at room temperature. From equation (2.8), the conductance of the saline within the spherical shell when its radius $l \to \infty$ is

$$\lim_{l\to\infty} G = 4\sigma\pi a \tag{2.9}$$

The resistance of the saline within the spherical shell on the outside of the electrode surface is

$$R = \frac{1}{G} \tag{2.10}$$

When the radius of the spherical shell $l \to \infty$, the resistance of the saline within it is given by equation (2.9) and (2.10)

$$\lim_{l\to\infty} R = \lim_{l\to\infty} \frac{1}{G} = \frac{1}{4\sigma\pi a} = \frac{\rho}{4\pi a} \tag{2.11}$$

which corresponds to equation (2.6). The uninsulated tip of the needle in the experiment in [5] in saline is solid and has a conical shape. The surface area of the tip is $0.3\, mm^2 = 0.3 \cdot 10^{-6}\, m^2$. Using this surface area, the spherical shaped electrode in figure (2.3) has the radius

$$a = \sqrt{\frac{0.3 \cdot 10^{-6}}{4\pi}}\, m \approx 1.55 \cdot 10^{-4}\, m \tag{2.12}$$

Let the value of the radius a be given by equation (2.12) and let $l = L/2 = 2.57\, mm = 2.57 \cdot 10^{-3}\, m$. With $n = 6$ in equation (2.8) (terms for $n > 6$ are insignificantly small and can be neglected)

$$\frac{R(n=6)}{\lim_{l\to\infty} R} = 0.97 \tag{2.13}$$

which means that 97 percent of the total resistance R in the saline is within the sensitivity volume with the radius $l = 2.57\,mm$.

As already described in Chapter 1 in this thesis, the result from the experiment in saline in [5] suggest that about 97 percent of the impedance modulus $|z|$ is covered by a spherical sensitivity volume with a radius of 2.6 mm from the tip of the needle. The article draws the conclusion that the major part of $|z|$ is within a spherical sensitivity volume with a radius between 2 mm and 3 mm for the needle type used in the experiment. The result of the Finite Element Analysis in [7], which has already been referred to in Chapter 1, is a spherical sensitivity volume with a radius of 3.75 mm from the tip of the needle. The sensitivity volume covers 97 percent of $|z|$ in the saline. Although the analytical model with a spherically shaped electrode with the same surface area as the needle ($0.3\,mm^2$) is a very simplified version of the experiment in saline and the Finite Element Model of the experiment, the result from the analytical model still fits well with the results from the experiment and the Finite Element Analysis.

Chapter 3

Test of instruments and needle electrodes

3.1 Introduction

Prior to the electrical impedance measurements with two different needle electrode types in fat and muscle tissue in a newly deceased pig at Rikshospitalet in Oslo, several test measurements were performed. The test measurements were performed in saline (0.9 percent NaCl), in fat and muscle tissue in a pork chop and in fat and muscle tissue in a piece of side meat. A pork chop was used in two tests, followed by two tests with a piece of side meat. A piece of side meat bought at the supermarket is thicker and has more fat tissue than a pork shop, so it was easier to work with considering the measurements. Unlike side meat, pork chops are available i most grocery stores. Therefore a pork chop was used in the first two tests with measurements in fat and muscle tissue.

3.2 The instruments and the electrodes

In the experiments in [5], the Solartron 1260/1294 system was used in order to perform the impedance measurements with the needle electrodes. This system consists of an impedance interface (1294) connected to a frequency analyzer (1260), and a laptop with the necessary software. The system is stationed at Rikshospitalet in Oslo, and it is quite large and heavy. Considering size and weight the Sciospec ISX-3 impedance analyzer and the MFIA impedance analyzer from Zurich Instruments are far more practical. These two instruments can be carried by one person between different labs. A

laptop with necessary software is also required for these two instruments. In this chapter, tests performed with the Sciospec impedance analyzer and the Zurich Instrument's impedance analyzer are described. The Solartron 1260/1294 system is not used to perform impedance measurements in this thesis.

The conclusion of the tests is that the MFIA impedance analyzer from Zurich Instruments is a better option than the Sciospec ISX-3 analyzer for the impedance measurements in fat and muscle tissue in a newly deceased pig. The reason is that the range of the impedance measurements in the impedance spectra seems to be too large for the Sciospec instrument. Depending on the choice of a parameter called *Range* in the Sciospec software, the impedance spectra were disturbed by noise in either the lower part, the higher part or both of these parts of the frequency range.

Three needle electrode types were used during the tests. Their length and diameter are stated on their packaging. Their diameter are stated in G, which means *Needle Gauge*. The Gauge number gives the diameter for solid needles and the outer and inner diameter for hollow needles. A description of Gauge and a table which translates G to mm, can be found at Wikipedia.org, see [11]. The surface area of the uninsulated tip of the three needle types, which is the surface area of the electrode, is not stated on their packaging. The three needle types were:

1. Needle type 1: The Teca Disposable Monopolar Needle electrode. The needles of this type used were quite old, with expiration date on 31 October 2011. They were manufactured by VIASYS Healthcare, which doesn't seem to manufacture this needle type anymore. The needle is solid and the uninsulated tip of the needle is conical. [12] shows a magnified picture of the uninsulated needle tip, and states the surface area of the tip to be 0.28 mm^2. The needle length is 37 mm and the diameter is 28 G = 0.36 mm. The needle type was not used in the impedance measurements in [5]. In this thesis it has only been used in the testing described in this chapter, and not in the impedance measurements in fat an muscle tissue in a newly deceased pig.

2. Needle type 2: The Stimuplex A Insulated Needle electrode manufactured by B. Braun Melsungen AG. The needle is hollow. The uninsulated tip of the needle is bevel cut and the cut surface has two angles of inclination. The manufacturer doesn't state the surface area of the

uninsulated needle tip. [12] shows a magnified picture of the uninsulated needle tip and also gives a rough estimate of the surface area of the uninsulated tip based on geometrical measurements through a microscope. The estimated surface area is 0.7 mm^2. The needle length of the needles used was 50 mm and 100 mm. The diameter is 21G, which means an outer diameter of 0.82 mm and an inner diameter of 0.51 mm. This needle type was one of the two needle types used in the impedance measurements in [5].

3. Needle type 3: The SonoPlex Needle electrode manufactured by Pajunk. The needle is hollow. The uninsulated tip of the needle, which is the electrode, is bevel cut and the cut surface has two angles of inclination. The manufacturer doesn't state the surface area of the uninsulated needle tip. The needle length is 100 mm and the diameter is 21G, which means an outer diameter of 0.82 mm and an inner diameter of 0.51 mm. This needle type was not used in the impedance measurements in [5].

Needle type 2 (Stimuplex A) was one of the of the two needle types used in the impedance measurements in [5]. Disposable Monopolar Needle electrode manufactured by Medtronic Inc., was the other needle type used in the article. It is solid and has a conical tip. The length of the needle is 37 mm and the diameter is 0.33 mm. The electrode area is 0.3 mm^2. So the needle type is quite similar to needle type 1 (Teca) just described. It was the Medtronic needle which was used in the experiment with two electrodes in saline in [5], in order to find the radius of the sensitivity volume of the needle.

In [5] the Medtronic needles were preconditioned with the constant current $I = 1\ \mu A$ in 1 minute in saline before they were used in impedance measurements in fat and muscle tissue. The purpose was to make the impedance measurements stable. The Stimuplex A needles were not preconditioned before use. In this thesis the needles were preconditioned in a different way. Each needle was used in order to establish several impedance spectra in a row in saline, in the frequency range between 10 Hz and 1 MHz, until the impedance spectra became stable. This frequency range is the same as the frequency range used in order to establish the impedance spectra in fat and muscle tissue in a newly deceased pig. The spectra were displayed in the software of each instrument, so the stabilization process could be followed.

The hollow Stimuplex A and Sonoplex needles were filled with saline prior to

the preconditioning, and blown with air inside afterwards in order to remove the saline inside the needles. Between impedance measurements in fat tissue and impedance measurements in muscle tissue, the needles were rinsed in saline. The hollow needles were also flushed inside with saline, and blown with air inside afterwards in order to remove the remaining saline inside the needles. In order to limit the number of needles used during the testing, several needles were used in more than one test session, or test day. Needles which were used in more than one test session, were rinsed in fresh water at the end of the test session and returned to their packaging for storage until the next session. The hollow needles were also flushed inside with fresh water, and blown with air inside afterwards in order to remove the remaining water inside the needles.

Chapter 2 describes the setup with three electrodes and the impedance analyzer. Figure (2.2) gives an illustration of the setup. As described in chapter 2, the three electrodes were:

- The needle electrode, which was connected to the impedance analyser's Working Sense port in order to measure the voltage, and to the Working port in order for the needle to be a Current Carrying (CC) electrode.
- An Ambu BlueSensor Q Ag/AgCl ECG electrode was connected to the Reference port. The voltage was measured with respect to this electrode.
- An Ambu BlueSensor Q Ag/AgCl ECG electrode was connected to the Counter port in order for it to be a CC electrode.

Figure (3.1) and figure (3.2) show the plastic tank used for the saline together with the three electrodes. Figure (3.3) shows the electrode setup for impedance measurements in a pork chop. The needle is positioned in fat tissue. Figure (3.4) shows the electrode setup for impedance measurements in a piece of side meat. The needle is positioned in muscle tissue.

Figure 3.1: The Reference ECG electrode and the Counter ECG electrode in the plastic tank with saline.

Figure 3.2: The needle is supported by a plastic tube, with a small diameter, through the lid of the tank with saline. The needle is of type 1 (Teca).

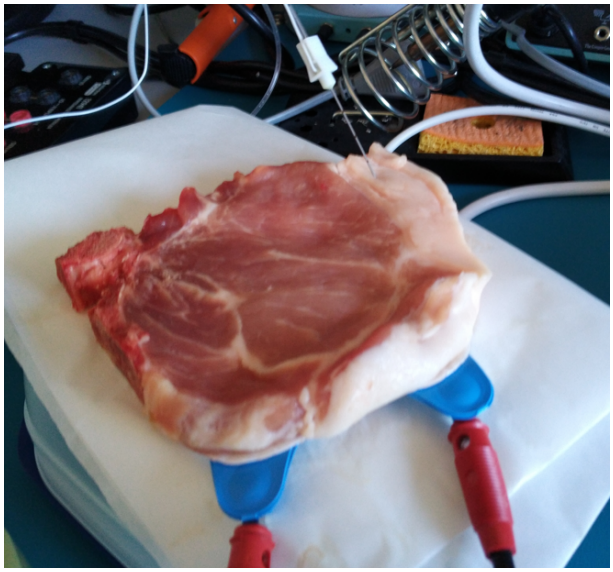

Figure 3.3: The needle positioned in fat tissue in a pork chop. The needle is of type 2 (Stimuplex A). The two ECG electrodes are positioned underneath the pork shop.

Figure 3.4: The needle positioned in muscle tissue in a piece of side meat. The needle is of type 1 (Teca). The two ECG electrodes are positioned on the skin underneath and out of sight at the back of the side meat.

3.3 The tests with Sciospec ISX-3

Three test sessions distributed on three different days were performed with the Sciospec ISX-3 impedance analyzer. In the first two tests, the impedance measurements were performed in saline. In the third test, the impedance measurements were performed in fat and muscle tissue in a pork shop. The conclusion is that the range of the impedance measurements in the impedance spectra seems to be too large for the instrument. Depending on the choice of a parameter called *Range* in the Sciospec software, the impedance spectra were disturbed by noise in either the lower part, the higher part or both of these parts of the frequency range. The noise are clearly visible in the impedance spectra. It can't be interpreted as Electrode Polarization Impedance (EPI), because EPI cant't be separated visually from the impedance in the saline or in the tissue.

Considering that it was necessary with many impedance spectra for the training and testing of the neural networks described in chapter 4 in this thesis, a test in session one shows that by extending the frequency range with one decade (1Hz to 1 MHz instead of 10 Hz to 1 MHz), the impedance measurements will take too long.

The impedance data files, created with the software, contained the real and the imaginary part of the impedance of all the measurement points in the impedance spectra. In the figures, the polar form of the impedance, with modulus $|Z|$ and phase angle θ, is shown. The basic information about the cartesian and the polar form of the impedance is covered in chapter 2.

3.3.1 Session one: First test in saline

During the first test, only a needle of type 1 (Teca) was applied. The needle was fresh from its sealed packaging. The following parameters were specified in the software for the first 20 impedance spectra

- Frequency range: From 10 Hz to 1 MHz.
- Peak voltage value: 30 mV.
- 30 measurement points on a logarithmic scale, which means 6 measurement points in each decade.
- Range: 100 Ω (Ohm).

- Precision: 0.5.

It is four different values which can be chosen for the Range parameter: 100 Ω, 10 kΩ, 100 kΩ and 1 MΩ. The Sciospec manual states that

- Range = 100 Ω is for impedance values up to 1 kΩ.
- Range = 10 kΩ is for impedance values up to 100 kΩ.
- Range = 1 MΩ is for impedance values greater than 100 kΩ.

The manual doesn't state anything for Range = 100 kΩ. The Precision parameter is a compromise between measurement precision and speed. Low precision settings correspond to fast measurements and lower accuracy. High precision settings correspond to greater accuracy, but longer measurement times. According to the manual, Precision = 0.5 gives good accuracy. With the given parameters, it took 15 seconds to establish an impedance spectrum in the frequency range between 10Hz and 1MHz. Since two data sets for the neural networks, each with around 200 impedance spectra from fat and muscle tissue in a newly deceased pig, were established through impedance measurements, it was important that it didn't take too much time to establish each spectrum.

Figure (3.5) shows impedance spectrum number 1, number 19 and number 20 of the first 20 impedance spectra. The figure shows that the three impedance spectra are disturbed by noise. It is mainly the phase angle which is influenced by the noise, and the noise occurs mainly in the lower part of the frequency range. Due to the noise, it is difficult to assess whether the needle has become stable as a result of the impedance measurements which established the first 20 impedance spectra.

In order to test the the four different values of the Range parameter, four additional impedance spectra were established. The frequency range was extended to also include the decade between 1 Hz and 10 Hz, without increasing the number of measurement points. The following parameters were then specified in the software for these impedance spectra

- Frequency range: From 1 Hz to 1 MHz.
- Peak voltage value: 30 mV.
- 30 measurement points on a logarithmic scale, which means 5 measurement points in each decade.

- Range: 100 Ω , 10 kΩ, 100 kΩ or 1 MΩ.
- Precision: 0.5.

With these parameters, it took 70 seconds to establish an impedance spectrum. This is too long time considering the establishment of the two data sets with impedance spectra from fat and muscle tissue in a newly deceased pig. Therefore the impedance range between 10 Hz and 1 MHz was used after this test.

Figure (3.6) shows the impedance spectra with the four different Range parameters. The impedance spectrum with Range = 1 MΩ is dominated by noise, and the impedance spectrum with Range = 100 kΩ is also heavily influenced by noise compared to the two impedance spectra with Range = 100 Ω and Range = 10 kΩ respectively. The following two test sessions will therefore only include Range = 100 Ω and Range = 10 kΩ.

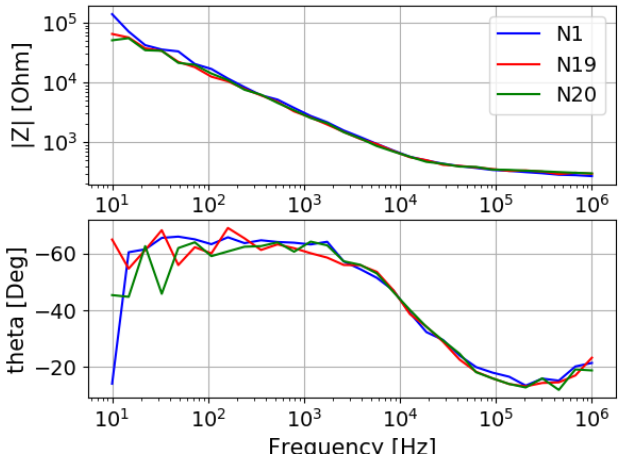

Figure 3.5: Impedance spectrum number 1, number 19 and number 20 of the first 20 impedance spectra in saline with a new needle of type 1 (Teca). Range parameter: 100Ω

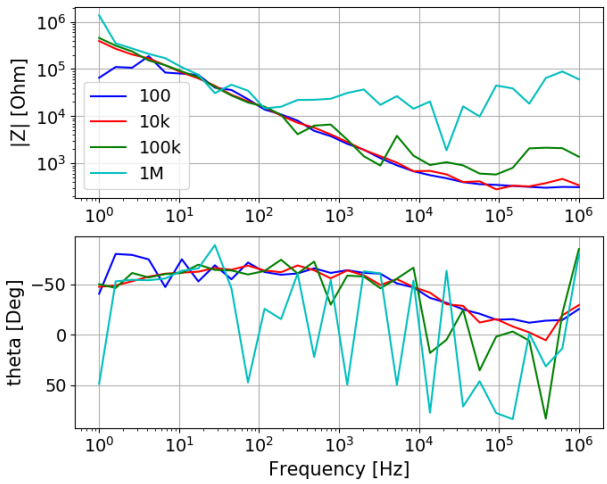

Figure 3.6: Four impedance spectra with the Teca needle. Range parameter: 100 Ω, 10 kΩ, 100 kΩ and 1 MΩ, respectively.

3.3.2 Session two: Second test in saline

The Teca needle used in the first test session was reused in this session. The second test session also included a needle of type 2 (Stimuplex A) and of type 3 (SonoPlex), fresh from their sealed packaging. Unfortunately the impedance spectra established with the SonoPlex needle showed unnaturally low impedance modulus. The error was discovered later, after the test session was completed. It turned out that the reason for the error was some sort of malfunction in the software. When unnaturally low $|Z|$ occurred in the spectra, it happened only under one more occasion ,the laptop connected to the Sciospec analyzer had to be restarted. Due to this error, the impedance spectra established with the SonoPlex needle are omitted here.

Since the first test session, two parameters were changed. The peak voltage value was increased from 30 mV to 50 mV. A peak value of 50 mV gives 35.4 mV rms (the root mean square of the peak voltage value), which is more in accordance with 30 mV rms used in the experiments in [5]. The number of measurement points were increased from 30 to 50. The increase in number of points will give the neural networks more data to work with. The increase in number of points also increased the time it took to establish an impedance spectrum, in the frequency range between 10 Hz and 1 MHz,

from 15 seconds to 25 seconds. The following parameters were then specified in the software

- Frequency range: From 10 Hz to 1 MHz.
- Peak voltage value: 50 mV.
- 50 measurement points on a logarithmic scale, which means 10 measurement points in each decade.
- Range = 100 Ω or Range = 10 kΩ.
- Precision: 0.5

Since the Teca needle was used in the first test session, just 10 impedance spectra in a row were established with Range = 10 kΩ, followed by a spectra with Range = 100 Ω. For the new Stimuplex A needle and SonoPlex needle, 15 impedance spectra in a row were established, followed by a spectrum with Range = 100 Ω.

Figure (3.7) shows impedance spectrum number 1, number 9 and number 10 with the Teca needle with Range = 10 kΩ. Range = 10 kΩ gives less noise in the lower part of the frequency range than Range = 100 Ω in figure (3.5). This is confirmed by figure (3.8), where spectrum number 10 with Range = 10 kΩ is compared with the spectrum with Range = 100 Ω. At the higher part of the frequency range, the two frequency spectra in figure (3.8) deviates from each other, but it is hard to assess which spectrum is affected by noise.

Figure (3.9) shows impedance spectrum number 1, number 14 and number 15 with the Stimuplex A needle with Range = 10 kΩ, and figure (3.10) compares spectrum number 15 with Range = 10 kΩ and the spectrum with Range = 100 Ω. Compared to the Teca needle, Range = 100 Ω gives less noise in the lower part of the frequency range. At the higher part of the frequency range, it is as for the Teca needle hard to assess which spectrum is affected by noise.

Due to less noise in the lower part of the frequency range with Range = 10 kΩ, figure (3.7) and figure (3.9) indicates that the impedance spectra gradually becomes more stable as more of them are established.

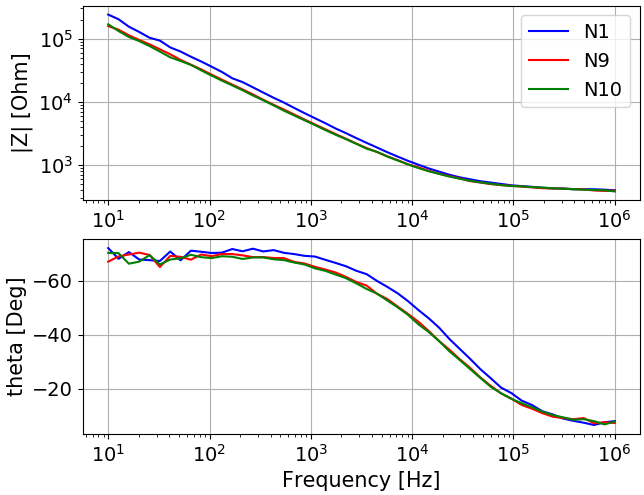

Figure 3.7: Impedance spectrum number 1, number 9 and number 10 in saline with the Teca needle. Range parameter: 10 kΩ

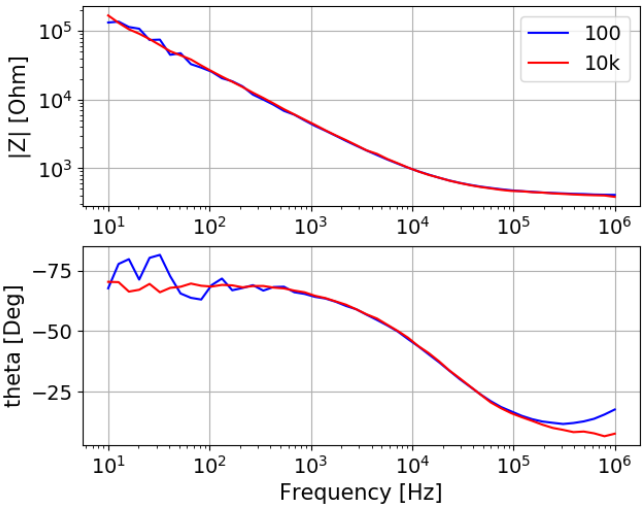

Figure 3.8: Impedance spectrum number 10 with Range = 10 kΩ and the spectrum with Range = 100 Ω with the Teca needle.

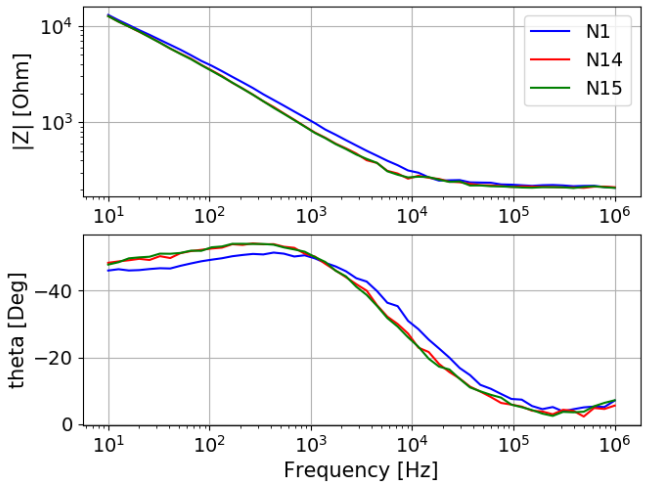

Figure 3.9: Impedance spectrum number 1, number 14 and number 15 in saline with the Stimuplex A needle. Range parameter: 10 kΩ

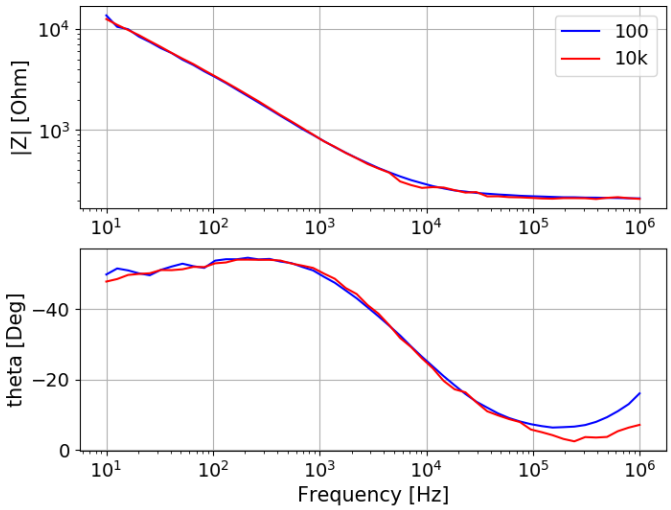

Figure 3.10: Impedance spectrum number 15 with Range = 10 kΩ and the spectrum with Range = 100 Ω with the Stimuplex A needle.

3.3.3 Session three: Impedance measurements in a pork chop

The three needles used in this test session were the same as the needles used in the second test session in saline. Since the needles had been used before to measure impedance in saline, they were considered to be stable and ready for impedance measurements in the fat and muscle tissue in the pork chop without any further measurements in saline prior to them. Figure (3.3) shows impedance measurements with a Stimuplex A needle (type 2) in the fat tissue in a pork chop.

The parameters specified in the software were the same as in the second test session in saline, see section 3.3.2. With the experience gained with the impedance measurements in saline, the main focus was on impedance measurements with Range = 10 kΩ. An impedance spectrum was also established with Range = 100 Ω with each of the three needles in the fat tissue in order to make a comparison, see figure (3.11) to figure (3.13). Each impedance spectrum with Range = 100 Ω was established right after four impedance spectra with Range = 10 kΩ. The needle position in the fat tissue was the same for these five spectra. Here is a summary of the measurement results, followed by six figures with impedance spectra

- The pattern of the unwanted noise is pretty similar to the noise pattern experienced during the second test session in saline. In the fat tissue Range = 100 Ω contributes to more noise than Range = 10 kΩ in the lower part of the frequency range. In the higher part of the frequency range, spectra with Range = 100 Ω and Range = 10 kΩ deviate from each other, but it is harder to assess the Range parameter which is the contributor to noise here.

- With Range = 10 kΩ, four impedance spectra established with the same needle in the same position in fat or muscle tissue are similar, except in the lower part of the frequency range where the unwanted noise is visible. It suggests that the needles are stable.

- The impedance spectra (Range = 10 kΩ) established with the same needle in two different positions in muscle tissue, don't coincide. For each needle, only one position in fat tissue was used.

- The impedance spectra (Range = 10 kΩ) in fat and muscle tissue, which are established with the same needle, are different. Since only one position in fat tissue and two positions in muscle tissue are used for each needle, it is to little information in order to make an assessment

whether there is any pattern which distinguishes the two tissue types from one another.

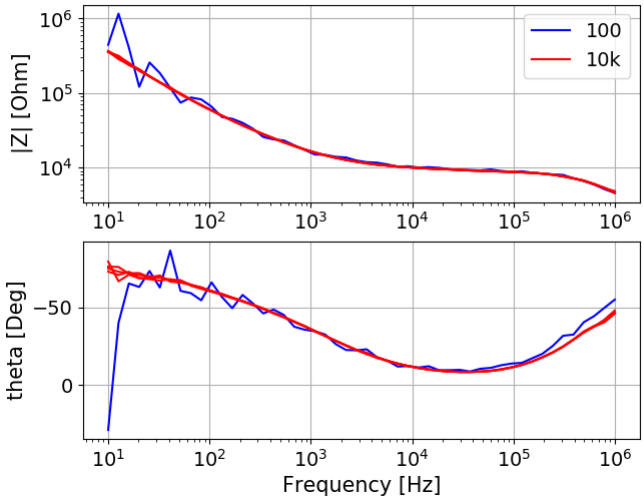

Figure 3.11: The impedance spectra with the Teca needle in fat tissue. Four spectra with Range = 10 kΩ and one spectrum with Range = 100 Ω.

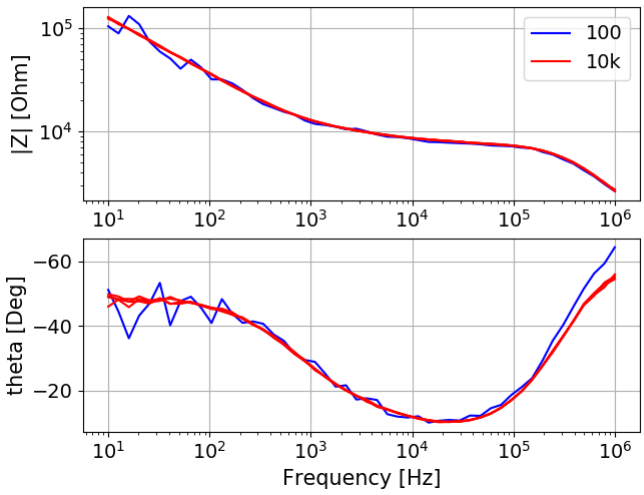

Figure 3.12: The impedance spectra with the Stimuplex A needle in fat tissue. Four spectra with Range = 10 kΩ and one spectrum with Range = 100 Ω.

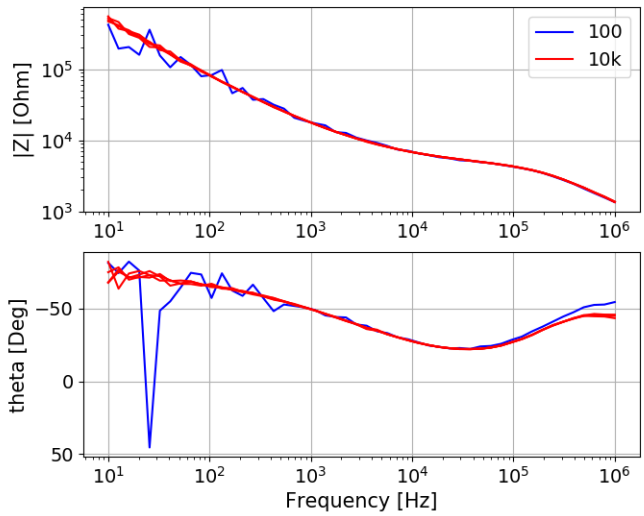

Figure 3.13: The impedance spectra with the SonoPlex needle in fat tissue. Four spectra with Range = 10 kΩ and one spectrum with Range = 100 Ω.

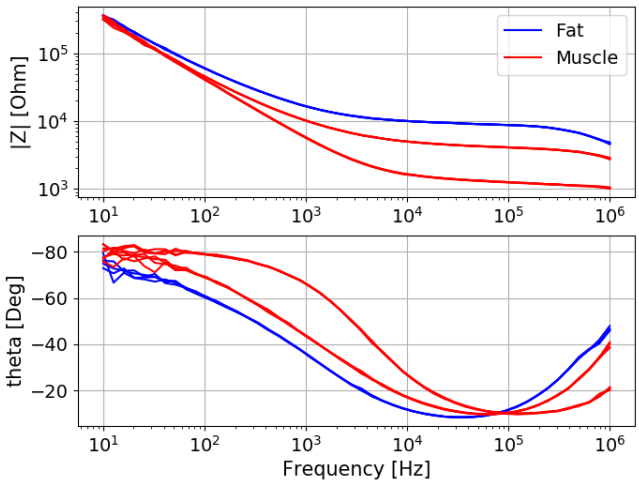

Figure 3.14: The impedance spectra with the Teca needle in one position in fat tissue and in two positions in muscle tissue. Four spectra in each position. Range = 10 kΩ.

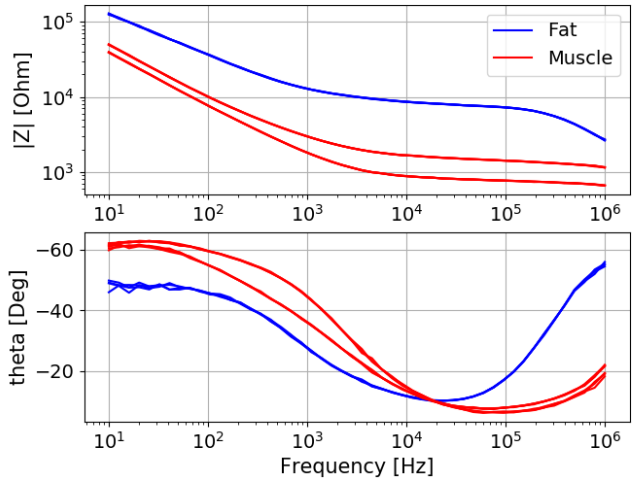

Figure 3.15: The impedance spectra with the Stimuplex A needle in one position in fat tissue and in two positions in muscle tissue. Four spectra in each position. Range = 10 kΩ.

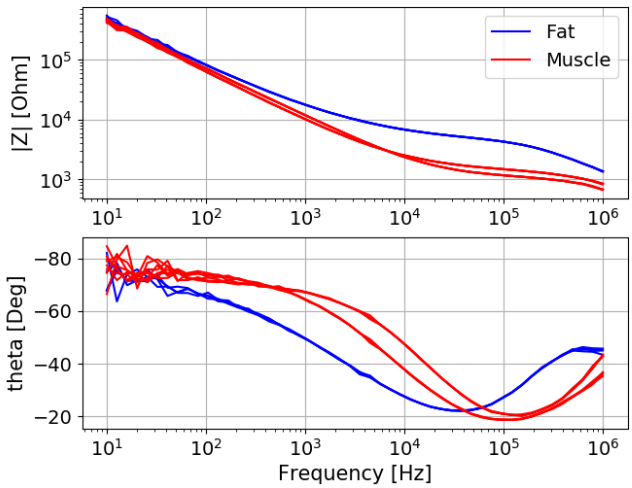

Figure 3.16: The impedance spectra with the SonoPlex needle in one position in fat tissue and in two positions in muscle tissue. Four spectra in each position. Range = 10 kΩ.

3.4 The tests with the Zurich impedance analyzer

Three test sessions distributed on three different days were performed with the MFIA impedance analyzer from Zurich Instruments. In the first test session, the impedance measurements were performed in saline. In the second test session, the impedance measurements were performed in fat and muscle tissue in both a pork chop and in a piece of side meat. Since there was no particular difference in the impedance spectra from the pork chop and from the piece of side meat, only the spectra from the side meat will be presented.

The impedance spectra in test session 1 and test session 2 were disturbed by noise in the measurement point at 51.8 Hz and at 104.8 Hz. The majority of the noise was in the measurement point at 51.8 Hz. 51.8 Hz is close to the frequency of the power grid, which is 50 Hz. 104.8 Hz is close to two times the frequency of the power grid. The source of this noise could be any electrical wire, piece of equipment at the laboratory or the power supply to the Zurich instrument itself. Turning off the fluorescent light tubes in the roof of the laboratory during the measurements didn't seem to have any

noticeable effect.

Prior to the third test session, two Sandberg Powerbank 20000 for Laptops were purchased. By using a powerbank (rechargeable battery) as the power source for the Zurich instrument instead of the power grid, the impedance analyzer should be excluded as a source of the noise. During the third test session, impedance measurements were performed in both saline and in fat and muscle tissue in a piece of side meat.

In addition to these three tests, a test where the needles were rinsed in ethanol for technical use was performed. The idea was that technical ethanol could be suitable for degreasing the needles and remove every trace of tissue between impedance measurements in the two different tissue types. The test revealed that the rinsing with technical ethanol caused changes in the impedance spectra from saline. The conclusion was therefore that the already established procedure with rinsing of the needles with saline between measurements in fat and muscle tissue was the most suitable.

The following parameters were specified in the Zurich Instrument's software, called LabOne, in all the test sessions

- Frequency range: From 10 Hz to 1 MHz.
- Peak voltage value: 50 mV.
- 50 measurement points on a logarithmic scale, which means 10 measurement points in each decade.

With the given parameters, it took 17 seconds to establish an impedance spectrum. There is no equivalent parameter to the Range parameter which has to be chosen in the LabOne software, so the Zurich instrument seems to be more usable when it comes to impedance spectra with a large range in impedance values. This is also evident in comparison with the impedance spectra established with the Sciospec instrument. The noise in the impedance spectra due to the choice of Range parameter in the Sciospec software has disappeared in the spectra established with the Zurich instrument.

The impedance data files, created with the LabOne software, contain the real part Z_x and the imaginary part Z_y of the impedance in all the measurement points in the impedance spectra. The polar form of the impedance spectra, with modulus $|Z|$ and phase angle θ, is shown in the figures. The basic

information about the cartesian and the polar form of the impedance is covered in chapter 2.

3.4.1 Session one: Test in saline

The three needles used were the same as the needles used in the last test with the Sciospec instrument. Even though the needles were already used, a new test in saline were performed because the Zurich instrument was used for the first time and because a control of the stability of the needles also could be useful after around two months of storage since the last test. After just a few impedance spectra were established, the needles were stable. The impedance spectra established with the Teca needle and the SonoPlex needle are disturbed by noise in the measurement point at 51.8 Hz. There is hardly any visible noise in the spectra established with the Stimuplex A needle. At the lower frequencies, the shape of the phase angle spectra established with the SonoPlex needle seemed odd. The shape deviated from the phase angle spectra established with the two other needles. Therefore a new SonoPlex needle, fresh from its sealed packaging, was also tested. The shape of the phase angle spectra established with the new SonoPlex needle agreed better with these spectra established with the Teca needle and the Stimuplex A needle. The old SonoPlex needle was therefore discarded. The impedance spectra established with the new SonoPlex needle are disturbed by some noise in the measurement point at 104.8 Hz in addition to the noise at 51.8 Hz. Figure (3.17) to figure (3.20) show the results for the four needles in the test session. The figures show that the needles were stable from impedance spectra number 6 and onward.

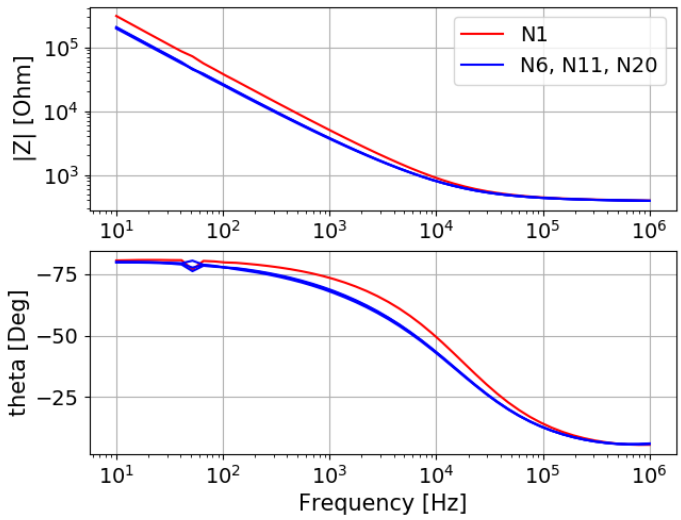

Figure 3.17: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the Teca needle.

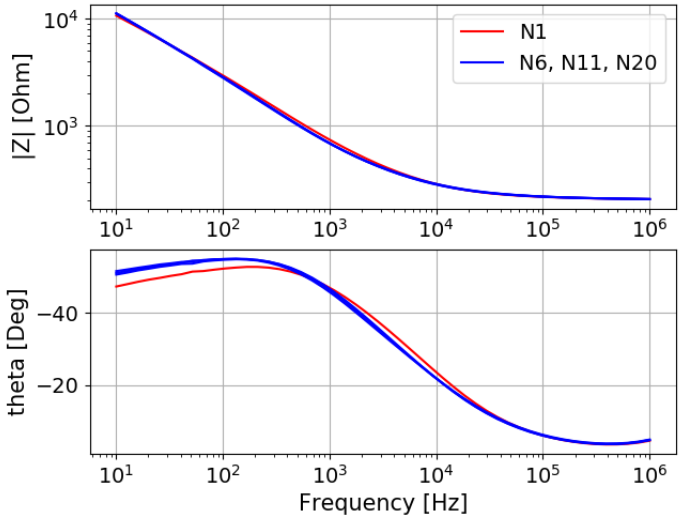

Figure 3.18: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the Stimuplex A needle.

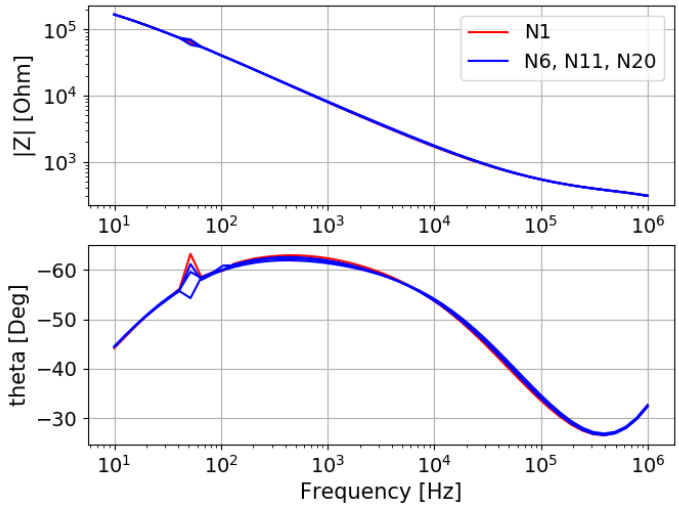

Figure 3.19: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the already used SonoPlex needle.

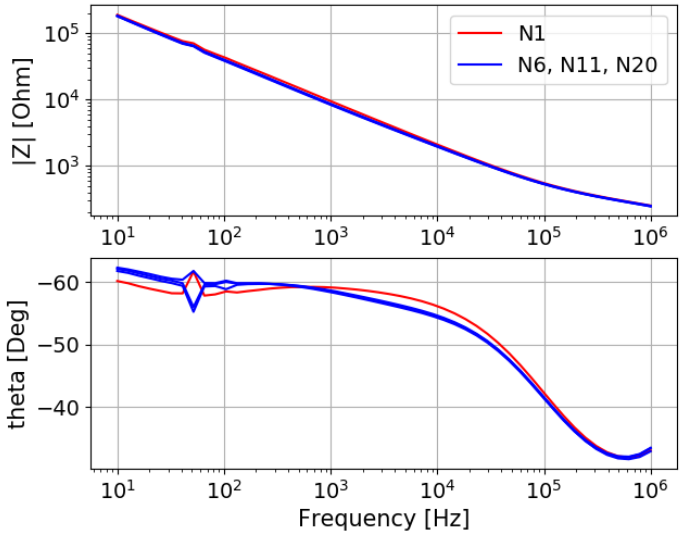

Figure 3.20: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the new SonoPlex needle.

3.4.2 Session two: Test in a piece of side meat

The three remaining needles in test session one in saline were also used in this test session. They were considered to be stable and ready for impedance measurements in the fat and muscle tissue in the side meat without any further measurements in saline prior to them. Figure (3.4) shows impedance measurements with a Teca needle in the muscle tissue in a piece of side meat.

For each of the three needles, five impedance spectra were established in two different positions in fat and in muscle tissue respectively. All the established spectra are present in the figures. They show that

- The five spectra established in the same needle position coincide. For the Stimuplex A needle, a slight difference between the spectra seems to occur some places.
- The impedance spectra are disturbed by noise in the measurement point at 51.8 Hz. The impedance spectra established with the Sono-Plex needle are in addition disturbed by some noise in the measurement point at 104.8 Hz.
- The impedance spectra established with the same needle in two different positions in one of the to tissue types, don't coincide.
- The impedance spectra in fat and muscle tissue respectively, which are established with the same needle, have different shape. With only two positions for each needle in fat and muscle tissue, respectively, it is to little information in order to make an assessment whether there is any pattern which distinguishes the two tissue types from one another.

Figure (3.21) to figure (3.23) show the results.

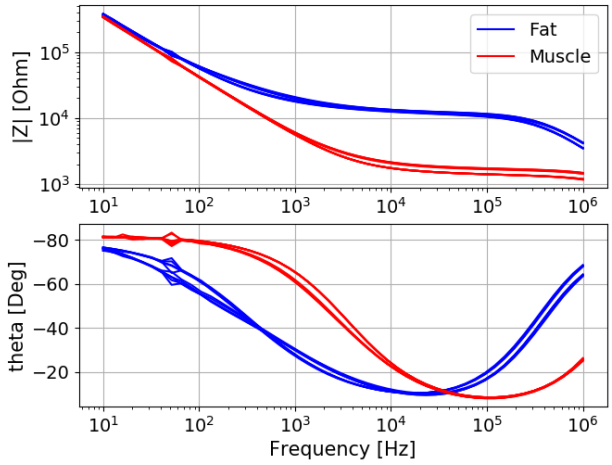

Figure 3.21: The impedance spectra with the Teca needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

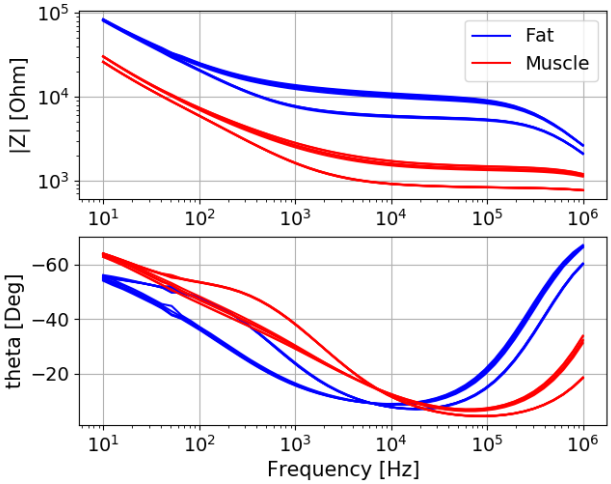

Figure 3.22: The impedance spectra with the Stimuplex A needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

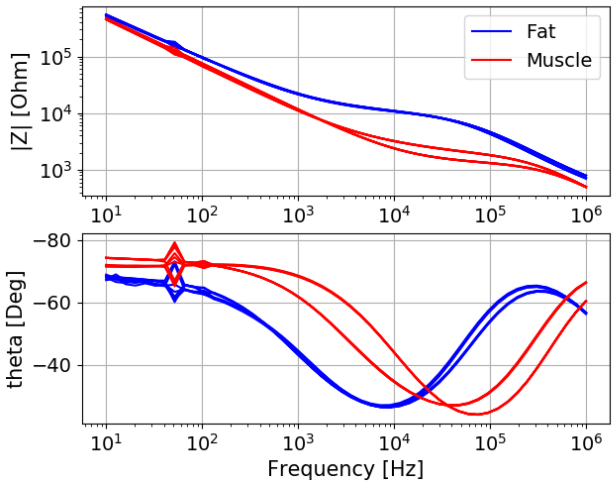

Figure 3.23: The impedance spectra with the SonoPlex needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

3.4.3 Session three: Test number two in a piece of side meat

In this test, two Sandberg Powerbank 20000 for Laptops were used as the power source for the Zurich instrument instead of the power grid. Each powerbank lasted just over 2 hours before it had to be replaced with the other powerbank, and recarged. All three needles used were new, fresh from their sealed packaging. As in the previous tests, one needle of each of the three needle types were used: A Teca needle, a Stimuplex A needle and a SonoPlex needle. Before the impedance measurements in the fat and muscle tissue in the side meat, each needle was preconditioned by establishing 20 impedance spectra in a row in saline. Four of the 20 impedance spectra are shown for each needle in figure (3.24) to figure (3.26). Compared to figure (3.17), the noise in the measurement point at 51.8 Hz has been reduced in the impedance spectra established with the Teca needle. In the impedance spectra established with the SonoPlex needle, there are not any visible noise at 51.8 Hz nor at 104.8 Hz anymore, which is a clear improvement from the spectra in figure (3.20).

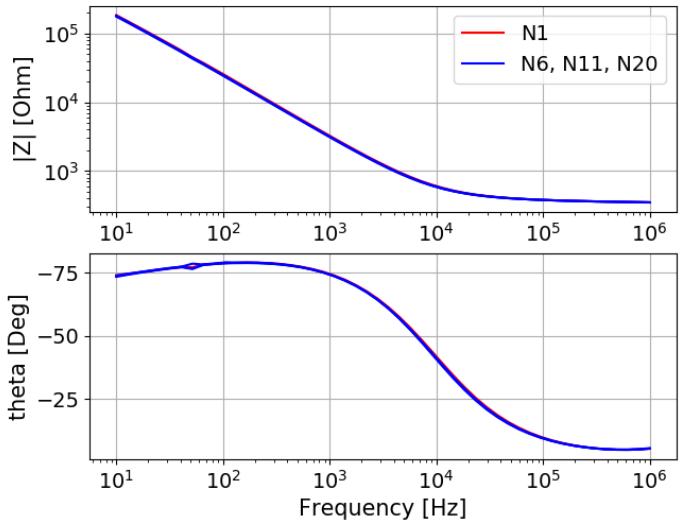

Figure 3.24: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the Teca needle.

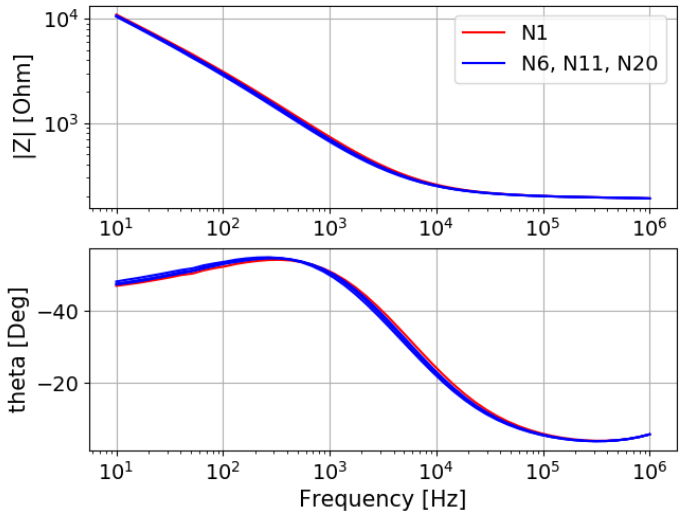

Figure 3.25: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the Stimuplex A needle.

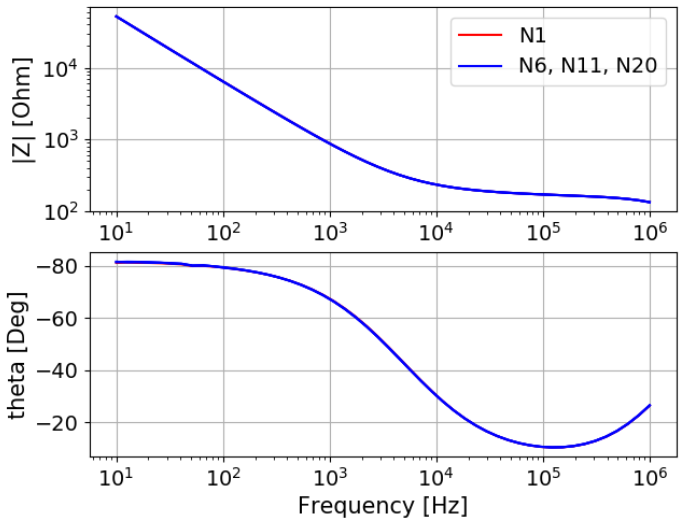

Figure 3.26: Impedance spectrum number 1, number 6, number 11 and number 20 in saline with the Sonoplex needle.

For each of the three needles, five impedance spectra were established in two different positions in fat and in muscle tissue respectively. All the established spectra are present in the figures. They show that

- For the SonoPlex needle, the five spectra established in the same needle position coincide. For the Stimuplex A needle, a slight difference between the spectra seems to occur some places. For the Teca needle it is a clear difference between spectra established in the same needle position, especially in the phase angle spectra.
- Compared to the impedance spectra in figure (3.21) to figure (3.23) established with the Zurich instrument's power supply from the power grid, the noise in the measurement point at 51.8 Hz is clearly reduced and the noise in the measurement point at 104.8 Hz seems to have disappeared.
- The impedance spectra in fat and muscle tissue respectively, which are established with the same needle, have different shape. With only two positions for each needle in fat and muscle tissue, respectively, it is to

little information in order to make an assessment whether there is any pattern which distinguishes the two tissue types from one another.

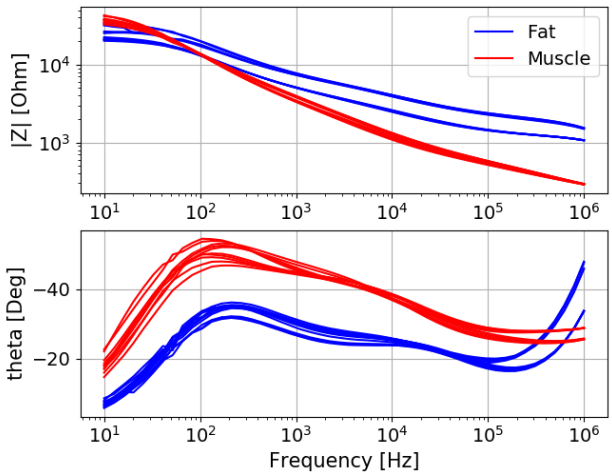

Figure 3.27: The impedance spectra with the Teca needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

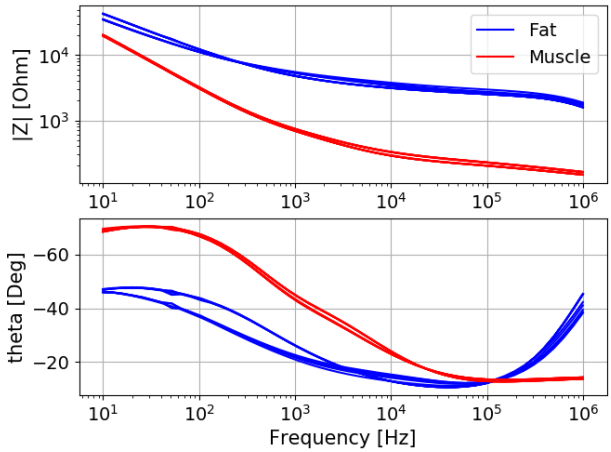

Figure 3.28: The impedance spectra with the Stimuplex A needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

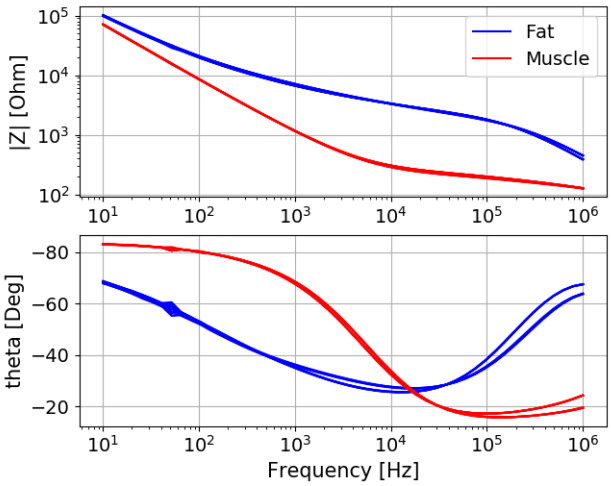

Figure 3.29: The impedance spectra with the SonoPlex needle in two positions in fat tissue and in two positions in muscle tissue. Five spectra in each position.

3.4.4 Additional test session with ethanol degreasing

In this test, a Sandberg Powerbank 20000 for Laptops were used as the power source for the Zurich instrument instead of the power grid. All three needles used in the test were new, fresh from their sealed packaging. As in the previous tests, one needle of each of the three needle types were used: A Teca needle, a Stimuplex A needle and a SonoPlex needle. Each needle was preconditioned by establishing 20 impedance spectra in a row in saline. After the preconditioning, the needles were rinsed in ethanol for technical use. The hollow Stimuplex A and Sonoplex needle were also flushed inside with technical ethanol, and blown with air inside afterwards in order to remove the remaining ethanol inside the needles. After the rinsing, another 20 impedance spectra in saline were established for each needle. The purpose of this additional test session was to investigate whether the rinsing with ethanol would affect the impedance measurements. The motivation for the test was the idea that technical ethanol could be suitable for degreasing the needles and remove every trace of tissue between impedance measurements in the two different tissue types. Figure (3.30) to figure (3.32) shows the last five impedance spectra, established prior to the rinsing with technical ethanol and after the rinsing. The impedance spectra are stable both prior to and after the rinsing, but the figures also reveal that the rinsing caused a change in the shape of the spectra established with the Teca needle and the Stimuplex A needle. The SonoPlex needle didn't seem to be affected by the rinsing. The conclusion was that the already established procedure with rinsing of the needles with saline between measurements in fat and muscle tissue was the most suitable approach.

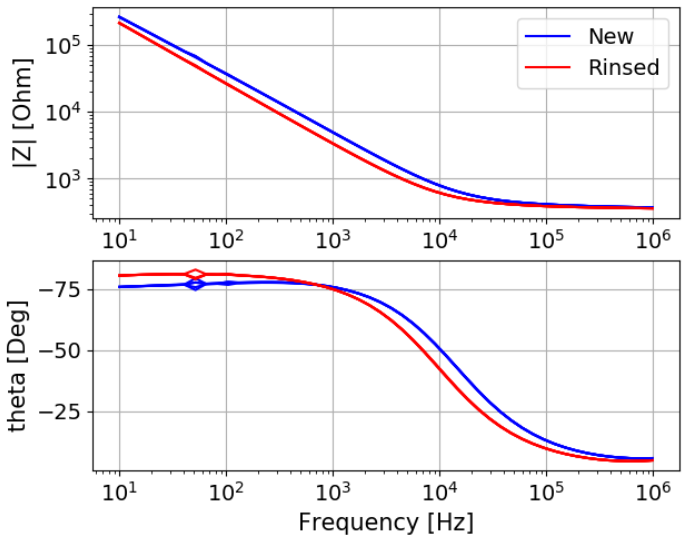

Figure 3.30: Five impedance spectra with the Teca needle prior to (New) and after the rinsing with technical ethanol.

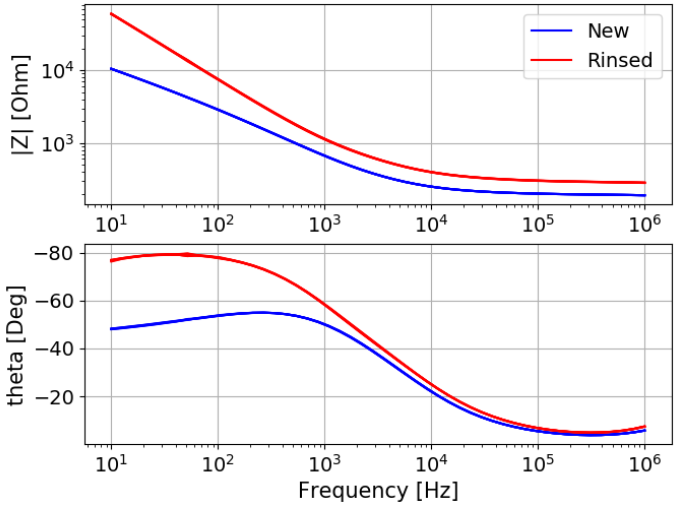

Figure 3.31: Five impedance spectra with the Stimuplex A needle prior to (New) and after the rinsing with technical ethanol.

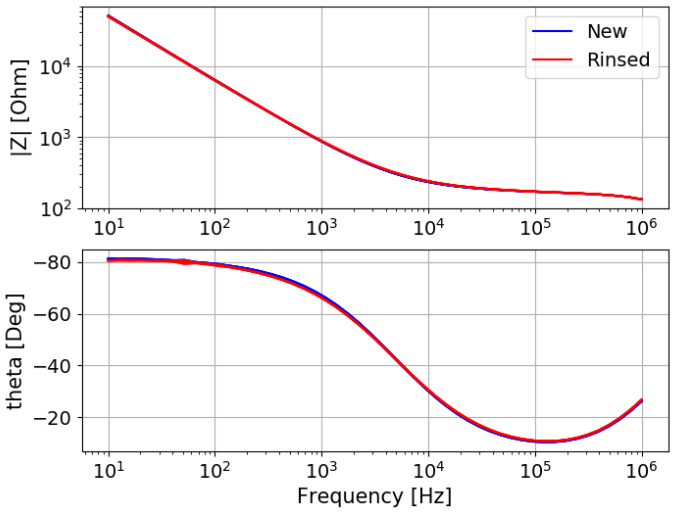

Figure 3.32: Five impedance spectra with the SonoPlex needle prior to (New) and after the rinsing with technical ethanol.

3.5 Test summary

Several tests have been performed with two different impedance analyzers and three different needle electrode types in saline, in two pork chops and in two pieces of side meat. Here is a summary and the conclusions of these tests, which were performed in preparation for the impedance measurements in fat and muscle tissue in a newly deceased pig at Rikshospitalet in Oslo.

- The range of the impedance measurements in the impedance spectra seems to be too large for the Sciospec instrument. Depending on the choice of the *Range* in the Sciospec software, the impedance spectra were disturbed by noise in either the lower part, the higher part or both of these parts of the frequency range. There is no equivalent parameter to the Range parameter which has to be chosen in the Zurich instrument's software called LabOne, so the Zurich instrument seems to be more usable when it comes to impedance spectra with a large range in impedance values. This is evident in comparison with the impedance spectra established with the Sciospec instrument. The noise in the impedance spectra due to the choice of Range parameter in the Sciospec software has disappeared in the spectra established with

the Zurich instrument. Therefore the Zurich instrument was chosen for the impedance measurements at Rikshospitalet.

- The impedance spectra established with the Zurich instrument were disturbed by noise in the measurement point at 51.8 Hz. There was also visible noise in the measurement point at 104.8 Hz in some of the spectra established with the SonoPlex needle. The power grid frequency is 50 Hz. By using a powerbank as the power source for the Zurich instrument instead of the power grid, the noise at 51.8 Hz was reduced, and the noise at 104.8 Hz vanished. Therefore two powerbanks were used as power supply for the Zurich instrument during the impedance measurements at Rikshospitalet. The second powerbank replaced the first powerbank when the first had to be recharged.

- The measurements at Rikshospitalet were performed during two afternoons/evenings in the fat and muscle tissue in a newly deceased pig. During each afternoon/evening, there was only time for measurements with one needle. Therefore two of the three needle types had to be chosen for these measurements. The Teca needles used during the tests were quite old, with expiration date on 31 October 2011. They were manufactured by VIASYS Healthcare, which doesn't seem to manufacture this needle type anymore. In the last test (session three with the Zurich instrument) in a piece of side meat, the Teca needle used didn't perform so well compared to the other two needles. Even though the needle became stable during the preconditioning in saline, see figure (3.24), there was a clear difference between spectra established in the same needle position in tissue afterwards, see figure (3.27). Therefore the Stimuplex A and the SonoPlex needle were chosen for the impedance measurements at Rikshospitalet.

The emphasis in this chapter has been on reducing the disturbance of visible noise in the impedance spectra, and the stability of the needles. The needles were considered to be stable when the impedance spectra in saline or in the same measurement point in fat or muscle tissue, in a pork chop or a piece of side meat, were stable. There hasn't been any emphasis on the fact that the spectra established by the different needle types don't coincide. In order for a comparison between impedance spectra established by the different needle types to be meaningful, the measurement conditions would have to be similar. New needles, fresh from their sealed packaging, have similar measurement conditions when they are preconditioned in saline. Then the stabilized impedance spectra from the preconditioning of the new needles

can be compared. The impedance measurements in section 3.4.3 and section 3.4.4 started with preconditioning of new needles in saline. In figure (3.33), the last impedance spectrum from the preconditioning of each of the three new needles from 3.4.4 are compared. The figure show that the spectra established with the different needle types are different. The three needle types are described in section 3.2. The Stimuplex A needle and the Sonoplex needle are both hollow and they have the same inner and outer diameter. Both needle types has an uninsulated tip (the electrode) which is bevel cut, and the cut surface has two angles of inclination. This information suggest that the shape and surface area of the uninsulated tip of these two needle types are quite similar. At the higher part of the frequency range, the modulus $|Z|$ are quite similar, but at the lower part of the frequency range their modulus is different. The difference modulus values at the lower part of the frequency range suggest different Electrode Polarization Impedance (EPI) here for the two needle types. The Teca needle is solid, and the uninsulated tip of the needle is conical. The needle diameter is less than half of the diameter of the two hollow needles, and the surface area of the uninsulated tip is less than half of the calculated surface area of the Stimuplex A's uninsulated tip. This information suggest that the Teca needle has a smaller sensitivity volume, see chapter 2 , which also leads to more EPI, see chapter 1. In the figure, the modulus $|Z|$ suggest that the level of EPI is higher for the Teca needle in the whole frequency range.

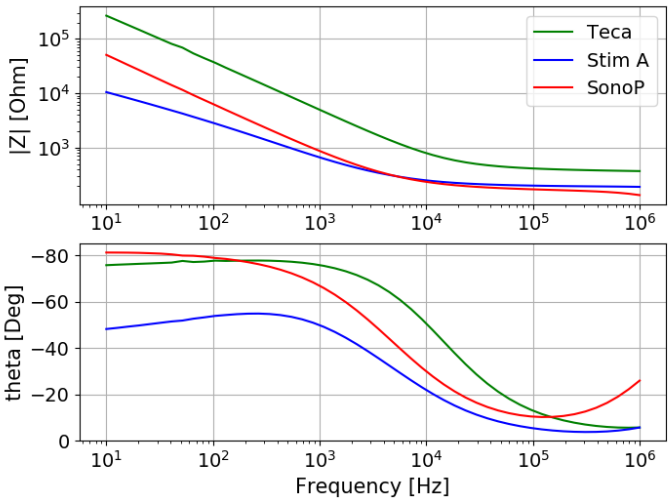

Figure 3.33: An impedance spectrum established in saline with a new and preconditioned Teca needle, Stimuplex A needle and SonoPlex needle respectively.

Chapter 4

The neural networks

4.1 Introduction

This chapter describes the Artificial Neural Networks (ANNs) which are applied in the analyzes of the two data sets with electrical impedance spectra established with impedance measurements with a needle electrode positioned in the fat and in the muscle tissue in a newly deceased pig. Each impedance spectrum contains 50 measurement points on a logarithmic scale, with 10 points in each of the 5 decades between 10 Hz and 1 MHz. The measurements were performed during two separate afternoons/evenings at Rikshospitalet in Oslo. For each of the two data sets, a separate type of needle electrode was used in order to measure the impedance. Chapter 1 gives an introduction to the impedance measurements, and 5 gives a more comprehensive description of these measurements and of the two data sets. The two needle types used are introduced in chapter 3.

ANNs are a type of machine learning. Applications of ANNs on impedance data has already shown promising results. In [13] ANNs are used to analyze data with simulated impedance spectra of liver ischemia in order to train the networks to predict the duration of the ischemia. This is an example of ANNs used on a regression problem: The prediction can have any value within a given time frame. In classification problems however, the outcome of an ANN is restricted to a limited set of values. In [14], ANNs are used to analyze impedance spectra from pigs under anesthesia in order to classify the grading of ischemic damage to the small intestine. In [15] ANNs are used on impedance spectra from stem cells in order to classify cell profileration from cell differentianion. In [13] and [14] two types of ANNs are used: Feed-

forward Neural Network (FNN) and Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells. In [15] only RNNs with LSTM cells are used. In this thesis both FNNs and a RNN with LSTM cells are used in the analysis of the impedance spectra in order to train the networks to classify the spectra from the tissue types as the right type of tissue. As in the three articles mentioned above, the learning is supervised, which means that the the correct tissue type are included for each impedance spectrum in the data set.

Tensorflow 2 and Keras are software packages used in order to build the ANNs for the classification of the impedance spectra. Tensorflow 2 is an open source platform for machine learning see https://www.tensorflow.org/. Keras, see https://keras.io/, is an API (Application Programming Interface) built on top of Tensorflow 2. Keras is easier to learn than Tensorflow 2 and makes much of the Tensorflow 2 functionality easier to implement, because the amount of programming technicalities are reduced. Some functionality from the Python machine learning library Scikit-Learn, see https://scikit-learn.org/stable/, will also be used. In addition, in order to get an idea about what is going on behind the scene in an ANN, a FNN will be programmed in Python, with some Scikit-Learn functionality added.

So what is the difference between a Feed-forward Neural Network (FNN) and a Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells. FNNs are used in order to analyze data sets with static data. If we want to classify the data examples in a data set, where the examples are simple digital images of handwritten numbers between 0 and 9, all the pixels of each image will enter the network simultaneously as an input array/vector. Therefore the order of the image pixels in the input vector doesn't matter as long as the pixel order is the same for all the images. It is similar for a data set with impedance spectra. For the FNN, the order of the impedance measurement values at the different frequencies doesn't matter as long as the order is the same in all the spectra.

Since all the data elements in a data example enter the FNN simultaneously, the network can't discover eventual patterns of time dependencies in the data. It is not so difficult to imagine some examples of dynamic data sets, where crucial information will be lost if time dependencies in the data are ignored. Examples can be from weather/climate data analyzes, economical data analyzes or data analyzes in order to build a model of an automated driving technology. Another example can be a model which is trained to

predict the next word in a text sequence. In order to predict the next word in a text sequence model, it is clearly important in which order the words are fed into the model. It is quite possible there is a time dependency in our impedance measurements as well. Therefore a Rerurrent Neural Network (RNN) with Long Short Term Memory (LSTM) cells, is also tested. RNN with LSTM cells is designed in order to analyze data sets with dynamic data.

In a RNN the data elements in the data set's data examples enters the network step by step in time steps. In the internal structure of the RNN, there are feedback loops which makes it possible for the network to have memory of the previous time steps when a new time step with data enters the network. The task of the LSTM cells in the internal structure of the RNN, is to learn to keep the memory of previous time steps in an optimal way, which means to learn what to keep and what to forget from previous steps in order to get the best learning outcome.

Even though Keras enables the user to make some simple FNNs and RNNs with LSTM cells after for example looking through some examples and copy parts of them, it might be a good idea to develop a basic understanding of ANNs first. In that way it is possible to get an understanding of the terminology used and a basic idea about what is going on behind the scene in an ANN. The knowledge can be acquired through attending a course in machine learning, reading the necessary parts of a textbook like [16] or finding information on the internet. Two websites which are helpful for a novice in machine learing are [17] and [19]. [17] explains the structure of a basic FNN, and also includes code examples in Python. The explanations can also be found in the eBook *An Introduction to Neural Networks for Beginners* which is included in the purchase of the eBook *Coding the Deep Learning Revolution* from the website, see [18]. *Coding the Deep Learning Revolution* introduces how to build both FNNs and a RNN with LSTM cells with Tensorflow 2 and Keras, but the emphasis is mainly on Tensorflow 2. Then [19] presents examples with fewer tecnicalities and is better in showing how Keras can make the implementation of ANNs easier.

4.2 A presentation of a FNN

In order to get an understanding of the terminology used and an idea about what is going on behind the scene in an ANN, a FNN is presented here. It

will also be programmed in python without the support of the Tensorflow 2 and Keras software packages, but with some functionality from the Python machine learning library Scikit-Learn included. The presentation of the FNN and the implementation in Python are based on [17].

Very simplified, the biological neural network in the brain is a huge network of interconnected neurons. One single neuron are connected to a large number of other neurons through synapses, which permits chemical signals to pass between the neurons. Learning occurs when certain parts of the network are activated repeatedly. The connections between the neurons in the network which cause the desired outcome are strengthened during this process. In an ANN, nodes represent the neurons in the brain. The connection between two nodes involves a weight and an activation function. As the training of the ANN proceeds, the weights are adjusted multiple times in order for the ANN to cause the desired training outcome.

Figure (4.1) shows the structure of a small neural network with an input layer (layer 1), an output layer (layer 4) and two hidden layers (layer 2 and layer 3). This is only an illustration. The input layer of this network is quite small. It can handle data examples with only 5 data elements as input. As an example, only 5 measuring points in an impedance spectrum will be too little information for a FNN in order to classify tissue types. The output layer size of the network is realistic though. A FNN with three output nodes can be used to classify input data examples in three different categories, for example impedance measurements from three tissue types or digital images of three different animal species, let's say cows, horses, and moose. The most simple neural networks have only one hidden layer, but here one extra hidden layer is added. Neural networks with two or more hidden layers are often referred to as deep neural networks. The number of hidden layers and the number of nodes in each hidden layer are two of several hyper-parameters of the network which need to be specified in order to optimize the network's learning outcome. The best way to specify hyper-parameters in general are by systematically experiment with combinations of them and compare the network's learning outcome for each combination. In addition the network include a bias node in the input layer and the two hidden layers. Their input value is 1. The transformation of the node values in the input layer as they pass through the hidden layers and end up in the output layer, take place when the values leave the nodes in a layer and travel along the straight lines in the figure to the nodes in the next layer. In the transformation, the node values are multiplied with weights and fed into an activation function.

In order to keep the figure clear and without clutter, some of the straight lines are omitted. The transformation of the input node values as they pass through the network in this way, is called the feed forward pass.

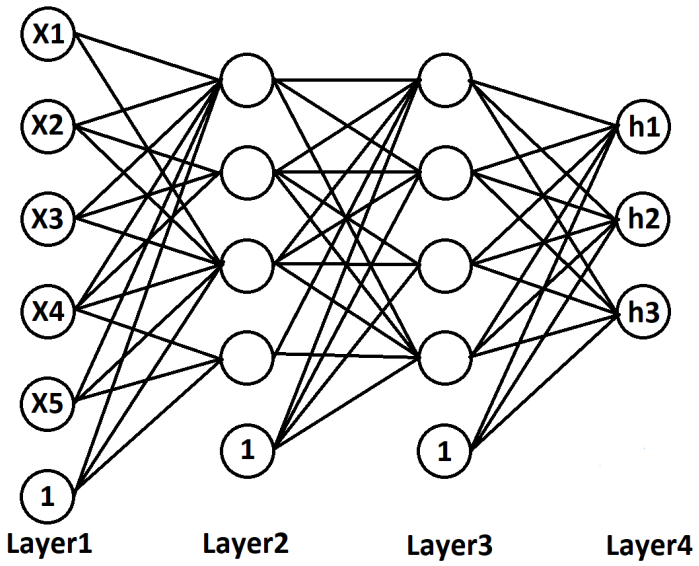

Figure 4.1: The structure of a neural network with two hidden layers. Some of the straight lines, which connects the nodes in a layer to all the nodes in the next layer, are omitted in order to keep a clear figure without clutter.

4.2.1 The feed forward pass

The simplest way to explain the feed forward pass is by using an illustration of a neural network as the one in figure (4.1), but first the organization of the input data is explained with some more details than has been done so far.

The data examples in a data set are organized as a matrix $\mathbf{X}_{data}$, where each row $\mathbf{x}$ is a data example. In a data set with digital images, each data example contains the pixels of a particular image. In a data set with impedance spectra in the frequency range between 10 Hz and 1 MHz, each data example contains a particular impedance spectrum. Each column in $\mathbf{X}_{data}$ is called a feature, which means that each particular pixel in an image

and each impedance value at a particular frequency is a feature. The training process of the network is supervised, which means that what each data example represent, is known in the training process of the network. What each data example represent is called the target y, which is an integer, for that example. If the data examples are digital images of cows, horses and moose, the three animal species, or the three targets, is represented by $y = 0$, $y = 1$ and $y = 2$ respectively in the training process. If the data examples are impedance spectra from fat and muscle tissue in a pig, the two tissue types, or the two targets, is represented by $y = 0$ and $y = 1$ respectively in the training process. A data example $\mathbf{x}$ and the corresponding target y of that example is called a pair. The vector $\mathbf{y}_{data}$ contains all the targets y. The data set then consist of the data examples in $\mathbf{X}_{data}$ and the corresponding targets in $\mathbf{y}_{data}$ Before the training of the network starts, the data set is divided into a training set with training pairs and a test set with test pairs. The test set is needed in order to control that the training has worked. If the result of the training is good, but the testing gives poor results, it indicates that the network only have been specialised in finding particular patterns in the training set, and not more general patterns for the type of data the whole data set is a representation of.

During training of the FNN, data examples from the training set with the elements, or the input node values, x_1 to x_5, pass through the network in figure (4.1). First, the input node values pass through to layer 2 and form the node values there. The transformation of the input values is performed in the following way:

$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} & w_{15}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} & w_{25}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} & w_{34}^{(1)} & w_{35}^{(1)} \\ w_{41}^{(1)} & w_{42}^{(1)} & w_{43}^{(1)} & w_{44}^{(1)} & w_{45}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix} \tag{4.1}$$

and

$$\begin{bmatrix} h_1^{(2)} \\ h_2^{(2)} \\ h_3^{(2)} \\ h_4^{(2)} \end{bmatrix} = f \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix} \tag{4.2}$$

In equation (4.1), $w_{ij}^{(l)}$ represent the weights which, together with the bias

weights $b_i^{(l)}$ and an activation function f, transform the input values in layer $l = 1$ in to the node values in layer $l + 1 = 2$. The index j represents the node number in layer $l = 1$ and the index i represents the node number in layer $l + 1 = 2$. In equation (4.2) the activation function f is applied on the outcome of equation (4.1), in order to form the node values in layer 2. In that way the weights and the activation function decides the influence a particular node will have in the neural network. Further, in the same way, the node values in layer 2 passes through to layer 3 and form the node values there:

$$\begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \\ z_3^{(3)} \\ z_4^{(3)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} & w_{14}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} & w_{24}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} & w_{34}^{(2)} \\ w_{41}^{(2)} & w_{42}^{(2)} & w_{43}^{(2)} & w_{44}^{(2)} \end{bmatrix} \begin{bmatrix} h_1^{(2)} \\ h_2^{(2)} \\ h_3^{(2)} \\ h_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \\ b_4^{(2)} \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} h_1^{(3)} \\ h_2^{(3)} \\ h_3^{(3)} \\ h_4^{(3)} \end{bmatrix} = f \begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \\ z_3^{(3)} \\ z_4^{(3)} \end{bmatrix} \tag{4.4}$$

And then the node values in layer 3 passes trough to layer 4 and form the output values:

$$\begin{bmatrix} z_1^{(4)} \\ z_2^{(4)} \\ z_3^{(4)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(3)} & w_{12}^{(3)} & w_{13}^{(3)} & w_{14}^{(3)} \\ w_{21}^{(3)} & w_{22}^{(3)} & w_{23}^{(3)} & w_{24}^{(3)} \\ w_{31}^{(3)} & w_{32}^{(3)} & w_{33}^{(3)} & w_{34}^{(3)} \end{bmatrix} \begin{bmatrix} h_1^{(3)} \\ h_2^{(3)} \\ h_3^{(3)} \\ h_4^{(3)} \end{bmatrix} + \begin{bmatrix} b_1^{(3)} \\ b_2^{(3)} \\ b_3^{(3)} \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} h_1^{(4)} \\ h_2^{(4)} \\ h_3^{(4)} \end{bmatrix} = f \begin{bmatrix} z_1^{(4)} \\ z_2^{(4)} \\ z_3^{(4)} \end{bmatrix} \tag{4.6}$$

A more compact form of equation (4.1) to (4.6) is

$$\begin{aligned} \mathbf{z}^{(l+1)} &= \mathbf{W}^{(l)}\mathbf{h}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{h}^{(l+1)} &= f\left(\mathbf{z}^{(l+1)}\right) \end{aligned} \tag{4.7}$$

where $l = 1, 2, \cdots, n-1$, with $n = 4$. The input node values from the data

example is $\mathbf{x} = \mathbf{h}^{(0)}$. The index form of equation (4.7) is

$$\begin{aligned} z_i^{(l+1)} &= \left(\sum_{j=1}^{S(l)} w_{ij}^{(l)} h_j^{(l)} \right) + b_i^{(l)} \\ h_i^{(l+1)} &= f\left(z_i^{(l+1)} \right) \end{aligned} \tag{4.8}$$

where $S(l)$ is the number of nodes in layer l.

A function called the sigmoid function or the logistic function is a commonly used activation function.

$$h = f(z) = \frac{1}{1 + e^{-z}}$$

Accoding to [16] at page 394, a scale parameter s can be included so that

$$h = f(z) = \frac{1}{1 + e^{-sz}} \tag{4.9}$$

With the scale parameter included, the derivative of the sigmoid function is

$$\begin{aligned} f'(z) &= s \frac{e^{-sz}}{(1 + e^{-sz})^2} \\ &= s \frac{1}{1 + e^{-sz}} \left(1 - \frac{1}{1 + e^{-sz}} \right) = s f(z)(1 - f(z)) \end{aligned} \tag{4.10}$$

The scale parameter s is also a hyper-parameter like the number of hidden layers and the number of nodes in each hidden layer. Figure (4.2) shows the sigmoid function and its derivative for three different scale parameters: $s = 1$, $s = 2$ and $s = 0.5$. The figure shows that the scale parameter decides how fast the function changes from values close to zero to values approaching one, and therefore how fast a node in the network can change from hardly any influence to its full influence potential on the network due to a change in the input z of the sigmoid function. The derivative is important when the values of the weights $w_{ij}^{(l)}$ and bias weights $b_i^{(l)}$ are updated during training of for the network. If the value of the derivative is too close to zero, the weight values doesn't change and the training of the network will slow down considerably or stop.

The weights $w_{ij}^{(l)}$ in the weight matrices $\mathbf{W}^{(l)}$ and the bias weights $b_i^{(l)}$ in the bias weight vectors $\mathbf{b}^{(l)}$ need some values to start with and will be

initialized with random values uniformly distributed in the interval $[0, 1\rangle$ in the Python programs of the FNN. This is not the most optimal initialization of these values, but for this FNN it is sufficient, and also a simple approach with respect to making the programs in Python without the support of the Tensorflow 2 and Keras software packages.

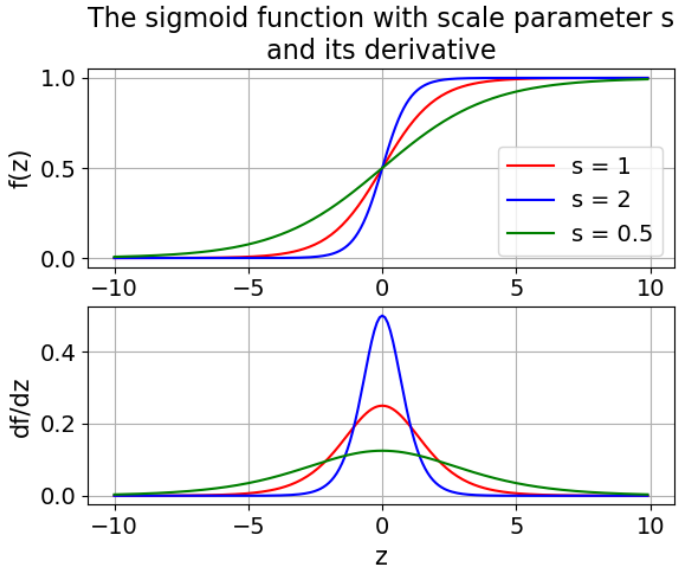

Figure 4.2: The sigmoid function with scale parameter $s = 1$, $s = 2$ and $s = 0.5$ and its derivative.

During training of the FNN, the output due to each data example passing through the network in the feed forward pass, is compared to the correct output, or the target, for that data example in a function called the *cost function* or the *loss function*. A FNN with three nodes in the output layer, like the FNN in figure (4.1), can be used to classify data sets containing data examples of three different categories, for example impedance spectra from three tissue types or digital images of three different animal species like for example cows, horses and moose. In order to be useful in the cost function, the targets need to be transformed from

$$\begin{aligned} y &= 0 \quad \text{for category 1} \\ y &= 1 \quad \text{for category 2} \\ y &= 2 \quad \text{for category 3} \end{aligned}$$

to

$$\begin{aligned}\mathbf{y} &= [1,0,0] \quad \text{for category 1}\\ \mathbf{y} &= [0,1,0] \quad \text{for category 2}\\ \mathbf{y} &= [0,0,1] \quad \text{for category 3}\end{aligned}$$

$\mathbf{y}$ is called the *one hot* vector representation of y. The cost function used here is simple. It calculates the L^2 norm of the difference between the output of a feed forward pass from a particular data example and the target for that data example. For a training pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is the input node values from the data example and $\mathbf{y}$ is the target, the cost function is expressed as

$$J = \frac{1}{2}\left\|\mathbf{y} - \mathbf{h}^{(n)}\right\|^2 = \frac{1}{2}\left\|y_i - h_i^{(n)}\right\|^2 \tag{4.11}$$

where $\mathbf{h}^{(n)}$ is the output from layer n, which is the output layer. For a FNN with four layers and three output nodes, like the network in figure (4.1), this cost function can be written as

$$\begin{aligned} J &= \frac{1}{2}\left\|\mathbf{y} - \mathbf{h}^{(4)}\right\|^2 = \frac{1}{2}\left\|y_i - h_i^{(4)}\right\|^2 \\ &= \frac{1}{2}\left(\sqrt{\left(y_1 - h_1^{(4)}\right)^2 + \left(y_2 - h_2^{(4)}\right)^2 + \left(y_3 - h_3^{(4)}\right)^2}\right)^2 \\ &= \frac{1}{2}\left(\left(y_1 - h_1^{(4)}\right)^2 + \left(y_2 - h_2^{(4)}\right)^2 + \left(y_3 - h_3^{(4)}\right)^2\right) \end{aligned} \tag{4.12}$$

In the process of training the network, the weight values are updated a large number of times. In order to update the weights, differentiation of the cost function is required. The $1/2$ fraction in front of the expressions makes the differentiation of the cost function more tidy.

4.2.2 Updating the weights: The gradient decent method with back propagation

The goal of the training of the FNN is to find the weights which minimize the value of the cost function J. In the training process, the weights in $\mathbf{W}^{(l)}$ and bias weights in $\mathbf{b}^{(l)}$ in layer $l = 1, 2, \cdots, n-1$, with $n = 4$ in a four layer FNN like the network in figure (4.1), are updated multiple times using a method called *the gradient decent method*. The method is iterative

and able to find a local minimum of a function which is differentiable. The gradient decent method for the updated weight and bias weight values are given in the following two expressions:

$$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha \frac{\partial J}{\partial \mathbf{W}^{(l)}} \tag{4.13}$$

and

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \frac{\partial J}{\partial \mathbf{b}^{(l)}} \tag{4.14}$$

for layer $l = 1, 2, \cdots, n-1$. The step size α is also a hyper-parameter which needs to be specified. The index form of the gradient decent expressions (4.13) and (4.14) are

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial J}{\partial w_{ij}^{(l)}} \tag{4.15}$$

and

$$b_{i}^{(l)} = b_{i}^{(l)} - \alpha \frac{\partial J}{\partial b_{i}^{(l)}} \tag{4.16}$$

Usually a batch of several training pairs randomly chosen from the training set are sent through the feed forward pass before the weights are updated. The gradient decent method are then expressed as

$$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha \left(\frac{1}{bs} \sum_{p=1}^{bs} \frac{\partial J}{\partial \mathbf{W}^{(l)}} \right) \tag{4.17}$$

and

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \left(\frac{1}{bs} \sum_{p=1}^{bs} \frac{\partial J}{\partial \mathbf{b}^{(l)}} \right) \tag{4.18}$$

where the batch size bs is the number of data pairs p in the batch. The batch size is also a hyper-parameter which has to be specified.

The gradient decent method with only one variable w can be used as an illustration of how the method works. Let the function J be a differentiable function with one independent variable w. Then the expression for the gradient decent method is

$$w_{new} = w_{old} - \alpha \frac{dJ}{dw} \tag{4.19}$$

Expression (4.19) shows that

- If J decends at $w = w_{old}$, then $w_{new} > w_{old}$. $w = w_{new}$ takes a step towards $w = w_{min}$ at the local minimum of J from the left.
- If J ascsends at $w = w_{old}$, then $w_{new} < w_{old}$. $w = w_{new}$ takes a step towards $w = w_{min}$ at the local minimum of J from the right.
- If the gradient magnitude is large at $w = w_{old}$, the step towards $w = w_{min}$ is larger compared to a similar step for a smaller gradient magnitude at $w = w_{old}$
- The step size parameter α decides how fast $w = w_{new}$ converges towards $w = w_{min}$. The parameter has to be tuned. If it is too large, $w = w_{new}$ will not converge towards $w = w_{min}$. If it is too small the convergence towards $w = w_{min}$ can be unnecessary slow or even stop.

As an example of the gradient decent method applied on a function with one independent variable, figure (4.3) shows the function $J(w) = w^4 - 3w^3 + 2$. The function has one inflection point at $w = 0$ and one minimum point at $w = 2.25$. $J(2.25) \approx -6.5430$ In the figure, the gradient decent method's start point is $J(w) = 6$ and the step size $\alpha = 0.08$. After 60 iteration steps the method has found $J(2.25) = -6.5430$. If the step size increases to $\alpha = 0.1$, w_{new} still fluctuates around w_{min} after 1000 iterations. If the step size decreases to $\alpha = 0.07$, w_{new} still hasn't been able to pass the inflection point after 1000 iterations.

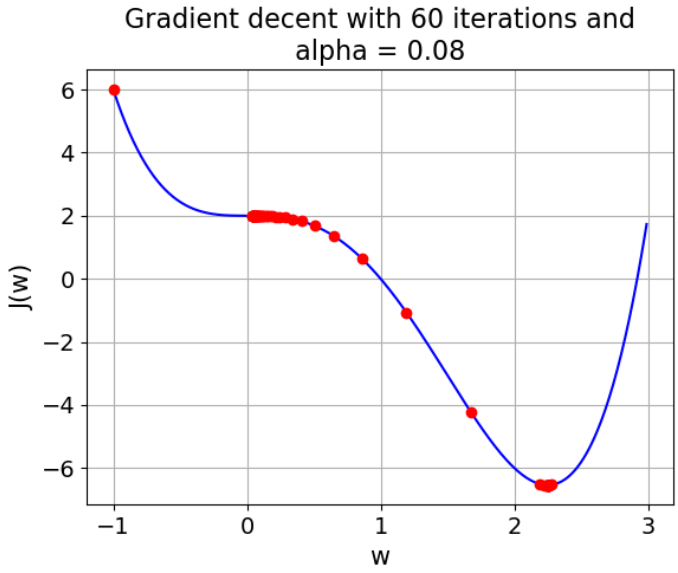

Figure 4.3: The gradient decent method applied on $J(w) = w^4 - 3w^3 + 2$ with 60 iterations. The start point for the method is $J(-1) = 6$ and the step size is $\alpha = 0.08$.

In order to update the weights $w_{ij}^{(l)}$ in equation (4.15) and the bias weights $b_i^{(l)}$ in equation (4.16), it is necessary to find an expression of $\partial J/\partial w_{ij}^{(l)}$ and $\partial J/\partial b_i^{(l)}$ respectively. The cost function J is dependent of the output $h_i^{(n)}$ of the network, see equation (4.11). $h_i^{(n)}$ is dependent of $z_i^{(n)}$, which again is dependent of $w_{ij}^{(n-1)}$ and $b_i^{(n-1)}$, see equation (4.8). Therefore

$$\frac{\partial J}{\partial w_{ij}^{(n-1)}} = \frac{\partial J}{\partial h_i^{(n)}} \frac{\partial h_i^{(n)}}{\partial z_i^{(n)}} \frac{\partial z_i^{(n)}}{\partial w_{ij}^{(n-1)}} \tag{4.20}$$

and

$$\frac{\partial J}{\partial b_i^{(n-1)}} = \frac{\partial J}{\partial h_i^{(n)}} \frac{\partial h_i^{(n)}}{\partial z_i^{(n)}} \frac{\partial z_i^{(n)}}{\partial b_i^{(n-1)}} \tag{4.21}$$

Since J only has $h_i^{(n)}$ as the independent variables, and not $h_i^{(l)}$ for layer $l = 2, \cdots, n-1$, it is not possible to find $\partial J/\partial w_{ij}^{(l-1)}$ and $\partial J/\partial b_i^{(l-1)}$ for $l = 2, \cdots, n-1$ directly. In order to solve this problem a method called *back propagation* will be introduced, but first we will take a closer look at

the partial derivatives in equation (4.20) and (4.21). Equation (4.12) shows the cost function J for the FNN in figure (4.1) with four layers and three output nodes in the output layer. For this cost function

$$\frac{\partial J}{\partial h_i^{(4)}} = -\left(y_i - h_i^{(4)}\right), \quad i = 1, 2, 3$$

It is easy to generalize this derivative for a FNN with n layers and two or more output nodes by a minor change in notation

$$\frac{\partial J}{\partial h_i^{(n)}} = -\left(y_i - h_i^{(n)}\right), \quad i = 1, 2, \cdots \tag{4.22}$$

Equation (4.8), the sigmoid function (4.9) and the derivative of the sigmoid function (4.10) give

$$\frac{\partial h_i^{(n)}}{\partial z_i^{(n)}} = f'\left(z_i^{(n)}\right) = sf\left(z_i^{(n)}\right)\left(1 - f\left(z_i^{(n)}\right)\right) \tag{4.23}$$

Equation (4.8) also gives

$$\frac{\partial z_i^{(n)}}{\partial w_{ij}^{(n-1)}} = h_j^{(n-1)} \tag{4.24}$$

and

$$\frac{\partial z_i^{(n)}}{\partial b_i^{(n-1)}} = 1 \tag{4.25}$$

Then (4.20) and (4.21) become

$$\frac{\partial J}{\partial w_{ij}^{(n-1)}} = -\left(y_i - h_i^{(n)}\right) f'\left(z_i^{(n)}\right) h_j^{(n-1)} \tag{4.26}$$

and

$$\frac{\partial J}{\partial b_i^{(n-1)}} = -\left(y_i - h_i^{(n)}\right) f'\left(z_i^{(n)}\right) \tag{4.27}$$

respectively, with the derivative $f'\left(z_i^{(n)}\right)$ of the sigmoid function given in (4.23). Introducing

$$\delta_i^{(n)} = -\left(y_i - h_i^{(n)}\right) f'\left(z_i^{(n)}\right) \tag{4.28}$$

equation (4.26) and (4.27) become

$$\frac{\partial J}{\partial w_{ij}^{(n-1)}} = \delta_i^{(n)} h_j^{(n-1)} \tag{4.29}$$

and

$$\frac{\partial J}{\partial b_i^{(n-1)}} = \delta_i^{(n)} \tag{4.30}$$

respectively.

In order to find $\partial J/\partial w_{ij}^{(l-1)}$ and $\partial J/\partial b_i^{(l-1)}$ for layer $l = n-1, \cdots, 3, 2$, the back propagation method is applied. The method solves the problem with the term $\partial J/\partial h_i^{(l)}$ in

$$\frac{\partial J}{\partial w_{ij}^{(l-1)}} = \frac{\partial J}{\partial h_i^{(l)}} \frac{\partial h_i^{(l)}}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l-1)}}$$

and

$$\frac{\partial J}{\partial b_i^{(l-1)}} = \frac{\partial J}{\partial h_i^{(l)}} \frac{\partial h_i^{(l)}}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial b_i^{(l-1)}}$$

for these layers, which can't be calculated directly since the cost function J is dependent of $h_i^{(n)}$ only. Going backwards through the layers from the output layer, the back propagation method is expressed as

$$\delta_j^{(l)} = \left(\sum_{i=1}^{S(l+1)} w_{ij}^{(l)} \delta_i^{(l+1)} \right) f'\left(z_j^{(l)}\right) \tag{4.31}$$

for layer $l = (n-1), \cdots, 3, 2$, and where $S(l+1)$ is the number of nodes in layer $l+1$.

Equation (4.29) and (4.30) can now be generalized to

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \delta_i^{(l+1)} h_j^{(l)} \tag{4.32}$$

and

$$\frac{\partial J}{\partial b_i^{(l)}} = \delta_i^{(l+1)} \tag{4.33}$$

for $l = n-1, \cdots, 2, 1$, where $\delta_i^{(n)}$ is given by equation (4.28) and $\delta_i^{(n-1)}, \cdots, \delta_i^{(3)}, \delta_i^{(2)}$ are given by equation (4.31). $h_j^{(1)}$ is the input node values x_j

4.2.3 Equations on matrix form

So far a mixture of matrix form and index notation have been used in order to explain the equations in the feed forward pass and in the updating of the weights with gradient decent, which included the back propagation method, in a FNN. In order for programs in Python to be as time efficient as possible, for loops needed for the implementation of equations expressed with index notation should be avoided. Instead these equations are rewritten to matrix form in order to utilize the Python library NumPy. The feed forward pass was explained using equations on matrix form , see equation (4.7) and the cost function in equation (4.11) and (4.12). In the gradient decent step, the matrix form of equation (4.28) becomes

$$\boldsymbol{\delta}^{(n)} = -\left(\mathbf{y} - \mathbf{h}^{(n)}\right) \circ f'\left(\mathbf{z}^{(n)}\right) \tag{4.34}$$

where $\circ$ is the symbol of the element-wise product between two vectors or two matrices of the same dimensions. The matrix form of the back propagation method in equation (4.31) becomes

$$\boldsymbol{\delta}^{(l)} = \left((\mathbf{W}^{(l)})^T \boldsymbol{\delta}^{(l+1)}\right) \circ f'\left(\mathbf{z}^{(l)}\right) \tag{4.35}$$

for layer $l = (n-1), \cdots, 3, 2$. The matrix form of equation (4.32) and (4.33) become

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l+1)} \left(\mathbf{h}^{(l)}\right)^T \tag{4.36}$$

and

$$\frac{\partial J}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l+1)} \tag{4.37}$$

for $l = n-1, \cdots, 2, 1$, where $\boldsymbol{\delta}^{(n)}$ is given by equation (4.34) and $\boldsymbol{\delta}^{(n-1)}, \cdots, \boldsymbol{\delta}^{(3)}, \boldsymbol{\delta}^{(2)}$ are given by equation (4.35). The row vector $\left(\mathbf{h}^{(l)}\right)^T$ is the transpose of the column vector $\mathbf{h}^{(l)}$.

4.2.4 An outline of the implementation of the FNN

We now have the equations we need in order to make an outline of the implementation of the FNN. The FNN will be implemented in Python and supported by some functionality from the Python machine learning library Scikit-Learn, see https://scikit-learn.org/stable/.

- The data set is divided into a training set with training pairs and a test set with test pairs. This task is performed by a Scikit-Learn function called *train_test_split*:

  from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

  splits the data set $(\mathbf{X}_{data}, \mathbf{y}_{data})$ into a training set $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ with training pairs and a test set $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ with test pairs. The test set here consists of 30 percent of the pairs in the data set. The training pairs and the test pairs are randomly chosen from the pairs in the data set. The random_state = 1 fixes the seed integer value to 1 in the pseudo-random number generator used to randomly chose the training set and test set. Then the same pairs from the data set end up in the training set and test set respectively each time. This is important in order to be able to compare results from different runs of the programs.

- As explained in conjunction with the cost function in equation (4.11): In order to be able to perform calculations with the cost function, each target y in $\mathbf{y}_{train}$ needs to be transformed to its corresponding *one hot* vector representation $\mathbf{y}$.

- A feature has already been defined as a column in the data set matrix $\mathbf{X}_{data}$. In order for all the features to be equally important in the training of the network, each feature in the training set matrix $\mathbf{X}_{\text{train}}$ are scaled by subtracting the mean of the feature from the feature itself, and dividing by the standard deviation of the feature. Then each feature in $\mathbf{X}_{\text{train}}$ is scaled and has a mean value equal to 0 and variance equal to 1. This task is performed by a Scikit-Learn function called *fit_transform*:

  from sklearn.preprocessing import StandardScaler
  X_scale = StandardScaler()

X_train = X_scale.fit_transform(X_train)

The features in the test set matrix $\mathbf{X}_{\text{test}}$ are not scaled in the same way as the features in the training set matrix. It is not desirable that the FNN possibly is able to recognize a scaling pattern in the test set features which is similar to the scaling of the training set features. Instead the mean and standard deviation values calculated for the features in the training set matrix are also applied to scale the features in the test set matrix. This task is performed by a Scikit-Learn function called *transform* just after the *fit_transform* function:

```
from sklearn.preprocessing import StandardScaler
X_scale = StandardScaler()
X_train = X_scale.fit_transform(X_train)
X_test = X_scale.transform(X_test)
```

- The structure of the FNN has to be decided. As an example, the network in figure (4.1) has an input layer with five input nodes, two hidden layers with four nodes each and a output layer with three output nodes. In addition a bias node is included in the input layer and in each of the two hidden layers. The bias nodes can be optional, but for simplicity they are a mandatory part of the structure here. The weight matrices and the bias weight vectors (which are implemented as arrays in Python) are stored in two separate Python dictionaries, one dictionary $W = \{\}$ for the weight matrices and one dictionary $b = \{\}$ for the bias weight vectors (arrays in Python). The structure of the network in figure (4.1) is described by the following Python list, which contains the number of nodes (not the bias nodes) in each layer: $lst = [5, 4, 4, 3]$. The dimensions of the weight matrices $\mathbf{W}^{(l)}$ and the bias weight vectors $\mathbf{b}^{(l)}$ (arrays in Python) then become

  $\mathbf{W}^{(1)} = W[1]$: a (4,5) matrix
  $\mathbf{W}^{(2)} = W[2]$: a (4,4) matrix
  $\mathbf{W}^{(3)} = W[3]$: a (3,4) matrix
  $\mathbf{b}^{(1)} = b[1]$: a (4,) array
  $\mathbf{b}^{(2)} = b[2]$: a (4,) array
  $\mathbf{b}^{(3)} = b[3]$: a (3,) array

As mentioned earlier in the chapter, the weights $w_{ij}^{(l)}$ in the weight matrices $\mathbf{W}^{(l)}$ and the bias weights $b_i^{(l)}$ in the bias weight vectors (arrays in Python) $\mathbf{b}^{(l)}$ will be initialized with random values uniformly distributed in the interval $[0, 1\rangle$.

- The number of data elements in the data examples decides the number of nodes in the input layer, while the number of categories in the data set to be classified decides the number of nodes in the output layer. The number of hidden layers and the number of nodes in each hidden layer are two of several hyper-parameters of the network which need to be specified in order to optimize the network's learning outcome. The other hyper-parameters introduced are the scale parameter s in the sigmoid function in equation (4.9), the step size α in the two gradient decent equations (4.17) and (4.18), and the number of training pairs in a batch, called the batch size bs. Usually a batch with more than one training pair randomly chosen from the training set are sent through the feed forward pass before the weights are updated with the gradient decent method. During one iteration, called an epoch, a batch is chosen and the weights are updated *(Number of pairs in the training set)//(batch size)* number of times. "//" is the operator for floor division in Python. Choosing a relatively large number of epochs, each training pair in the training set will take part in the training of the FNN in an approximately equal number of times.

- In order to calculate the sum in the two gradient decent equations (4.17) and (4.18), one Python dictionary $\Delta W = \{\}$ for storage of the matrices in the sum in equation (4.17) and one Python dictionary $\Delta b = \{\}$ for storage of the vectors (arrays in Python) in the sum in equation (4.18) are introduced. The matrices stored in ΔW and the vectors (arrays in Python) stored in Δb are initialized with values equal to zero before each batch of randomly chosen training pairs are sent through the feed forward pass. As an example, the network in figure (4.1) have matrices $\partial J/\partial \mathbf{W}^{(l)} = \Delta \mathbf{W}^{(l)}$ and vectors (arrays in Python) $\partial J/\partial \mathbf{b}^{(l)} = \Delta \mathbf{b}^{(l)}$ with the following dimensions which of course are the same dimensions as the network's weight matrices $\mathbf{W}^{(l)}$ and weight vectors (arrays) $\mathbf{b}^{(l)}$ have:

$\partial J/\partial \mathbf{W}^{(1)} = \Delta \mathbf{W}^{(1)} = \Delta W[1]:$ a (4,5) matrix
$\partial J/\partial \mathbf{W}^{(2)} = \Delta \mathbf{W}^{(2)} = \Delta W[2]:$ a (4,4) matrix
$\partial J/\partial \mathbf{W}^{(3)} = \Delta \mathbf{W}^{(3)} = \Delta W[3]:$ a (3,4) matrix

$\partial J/\partial \mathbf{b}^{(1)} = \Delta \mathbf{b}^{(1)} = \Delta b[1]$: a (4,) array
$\partial J/\partial \mathbf{b}^{(2)} = \Delta \mathbf{b}^{(2)} = \Delta b[2]$: a (4,) array
$\partial J/\partial \mathbf{b}^{(3)} = \Delta \mathbf{b}^{(3)} = \Delta b[3]$: a (3,) array

- In order to calculate $\partial J/\partial \mathbf{W}^{(l)}$ and $\partial J/\partial \mathbf{b}^{(l)}$ for each training pair in the batch, equation (4.36) and (4.37) are applied. These expressions involve $\boldsymbol{\delta}^{(n)}$ and $\boldsymbol{\delta}^{(l)}$ for layer $l = (n-1), \cdots, 3, 2$, which are given in equation (4.34) and (4.35) respectively. The $\boldsymbol{\delta}$ vectors (arrays in Python) for a training pair in the batch are stored temporarily in a Python dictionary $\delta = \{\}$, until the calculation of $\partial J/\partial \mathbf{W}^{(l)}$ and $\partial J/\partial \mathbf{b}^{(l)}$ are completed for a particular training pair. The dimensions of the $\boldsymbol{\delta}$ components in the network in figure (4.1) are:

  $\boldsymbol{\delta}^{(4)} = \delta[4]$: a (3,) array
  $\boldsymbol{\delta}^{(3)} = \delta[3]$: a (4,) array
  $\boldsymbol{\delta}^{(2)} = \delta[2]$: a (4,) array

- In order to keep track on the progress of the training of the FNN, the average of the cost J for all the training pairs which participate in the training during each epoch will be calculated. The cost function for one training pair is given in equation (4.11). The average cost for one epoch then is (*tp* means *training pair* in the sum below)

$$\hat{J} = \frac{1}{m} \sum_{tp=1}^{m} J = \frac{1}{2m} \sum_{tp=1}^{m} \left\| \mathbf{y} - \mathbf{h}^{(n)} \right\|^2 \tag{4.38}$$

  where m = (batch size) x (number of batches in the epoch) is the number of training pairs which participate in the training during each epoch.

- An outline of the implementation of the training part of the FNN can now look like this :

  Create training set and test set from the data set. Scale the features in the training set and test set.

  Decide the structure of the FNN and initialize the weight matrices $\mathbf{W}^{(l)}$ and bias weight vectors (arrays) $\mathbf{b}^{(l)}$.

Decide number of epochs, batch size bs, scale parameter s and step size α. The number of randomly chosen batches from the training set during one epoch is *nbs = (Number of pairs in the training set)//(batch size).*

count = 0
while count < number of epochs:
  #Comment: Initialize average cost variable for the epoch
  avg_cost = 0
  #Comment: for the number of batches nbs
  for $j = 0, \cdots, nbs-2, nbs-1$:
    Initialize the ΔW matrices and Δb arrays
    #Comment: bs is the number of training pairs in a batch
    Create batch of bs randomly chosen training pairs
    for each training pair in the batch:
      #Comment: Create a Python dictionary δ
      $\delta = \{\}$
      The feed forward pass, see equation (4.7)
      #Comment: n = number of layers in the FNN
      for $l = n, \cdots, 2, 1$:
        if l == n:
          δ[l] = see equation (4.34)
          avg_cost = avg_cost + equation (4.11)
        else:
          if l > 1:
            δ[l] = see equation (4.35)
          #Comment: Building the sum in equation (4.17) and (4.18)
          #Comment: The arrays $\delta^{(l+1)}$ and $h^{(l)}$ are transformed to
          #column vectors in the application of equation (4.36)
          ΔW[l] = ΔW[l] + equation (4.36)
          Δb[l] = Δb[l] + equation (4.37)
    #Comment: The gradient decent step for the training pairs
    #in the current batch with batch size bs
    for $l = n-1, \cdots, 2, 1$:
      #Comment: Equation (4.17)
      W[l] = W[l] - $(\alpha/bs) * \Delta$W[l]
      #Comment: Equation (4.18)
      b[l] = b[l] - $(\alpha/bs) * \Delta$b[l]
  #Comment: Calculation of average cost for the epoch. The
  #result can be printed and/or saved in a Python list for plotting

m = bs*nbs
avg_cost = (1/m)*avg_cost
count = count + 1

After each epoch, the progress of how accurate the FNN has become in classifying the categories in the data set is tested by applying the test set. The test examples passes through the feed forward pass, and the index i of the output node $h_i^{(n)}$ with the largest value for each test example is compared with the corresponding target in the test pair. The test accuracy is calculated as the number of test pairs with equal index values and corresponding target values, divided by the number of test pairs in the test set. The actual test accuracy is calculated by a Scikit-Learn function called *accuracy_score*.

4.2.5 An improvement of the gradient decent method through L2 regularization

This section is based on [20]. During training, the FNN can be a specialist on classifying the training examples in the training set. This specialization can involve fine tuning the weights so that also insignificant details and arbitrary noise are misinterpreted as an important part of the pattern in the training data. A network which is so fine tuned towards all the small details in the training data, misses the general patterns and might therefore perform poorly in classifying fresh data, like the test data, which has not taken part in the training of the network. It becomes difficult for the network to see the forest for the trees. It has been over-fitted to the training data

During training of the FNN, the weights $w_{ij}^{(l)}$ and bias weights $b_i^{(l)}$ are adjusted multiple times by applying the gradient decent method, in order for the value of the cost function J in equation (4.11) to converge towards a local minimum point of the function. The local minimum of J found by the gradient decent method is one of several local minimum values, and the value found might not be the one which make the FNN perform at its best. If the convergence process involves adjustments of the weights $w_{ij}^{(l)}$ towards large values, it is a sign of over-fitting of the network. With large weight values, small variations in the input like arbitrary noise might lead to an excessive impact on the network. This is because large weights lead to excessive amplifications of these small variations.

In order to avoid too large weights, the cost function J in equation (4.11)

can be adjusted to

$$J = J_1 + J_2 = \frac{1}{2}\left\|y_i - h_i^{(n)}\right\|^2 + \frac{\lambda}{2}\sum_{i,j}\left(w_{ij}^{(l)}\right)^2 \tag{4.39}$$

Since the sum of the squared values of the weights are calculated in J_2, the regularization is called *L2 regularization*. The regularization parameter λ is a hyper-parameter which value usually is quite small. Since the FNN adjust the weights to values which give convergence towards a local minimum of the cost function J, the additional part J_2 prevents the weights from growing too large. Now

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J_1}{\partial w_{ij}^{(l)}} + \frac{\partial J_2}{\partial w_{ij}^{(l)}} \tag{4.40}$$

for $l = n-1, \cdots, 2, 1$, where

$$\frac{\partial J_1}{\partial w_{ij}^{(l)}} = \delta_i^{(l+1)} h_j^{(l)}$$

is known already from equation (4.32). The second part becomes

$$\frac{\partial J_2}{\partial w_{ij}^{(l)}} = \frac{\lambda}{2}\frac{\partial}{\partial w_{ij}^{(l)}}\sum_{i,j}\left(w_{ij}^{(l)}\right)^2 = \lambda w_{ij}^{(l)} \tag{4.41}$$

It is not straight forward to see what is happening during the differentiation in equation (4.41). A simple example makes it more clear: Let

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

Then

$$\sum_{i,j}\left(w_{ij}^{(l)}\right)^2 = w_{11}^2 + w_{12}^2 + w_{21}^2 + w_{22}^2$$

and

$$
\begin{aligned}
&\frac{\partial}{\partial w_{ij}}\left(w_{11}^2+w_{12}^2+w_{21}^2+w_{22}^2\right)\\
&=2w_{11}\frac{\partial w_{11}}{\partial w_{ij}}+2w_{12}\frac{\partial w_{12}}{\partial w_{ij}}+2w_{21}\frac{\partial w_{21}}{\partial w_{ij}}+2w_{22}\frac{\partial w_{22}}{\partial w_{ij}}\\
&=2w_{11}\begin{bmatrix}1&0\\0&0\end{bmatrix}+2w_{12}\begin{bmatrix}0&1\\0&0\end{bmatrix}+2w_{21}\begin{bmatrix}0&0\\1&0\end{bmatrix}+2w_{22}\begin{bmatrix}0&0\\0&1\end{bmatrix}\\
&=2\begin{bmatrix}w_{11}&w_{12}\\w_{21}&w_{22}\end{bmatrix}=2w_{ij}=2\mathbf{W}
\end{aligned}
$$

Equation (4.32) can now be replaced with

$$\frac{\partial J}{\partial w_{ij}^{(l)}}=\delta_i^{(l+1)}h_j^{(l)}+\lambda w_{ij}^{(l)} \tag{4.42}$$

The matrix form of this equation is

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}}=\boldsymbol{\delta}^{(l+1)}\left(\mathbf{h}^{(l)}\right)^T+\lambda\mathbf{W}^{(l)} \tag{4.43}$$

which replaces equation (4.36).

In the outline of the implementation of the training of the FNN, any update of the matrices $\mathbf{W}^{(l)}$ don't take place before all the training pairs in a particular batch have passed trough the FNN. It is therefore only necessary to replace the gradient decent line

W[l] = W[l] - $(\alpha/bs) * \Delta$W[l]

which is the original gradient decent step in equation (4.17) for a batch, with

W[l] = W[l] - $\alpha((1/bs) * \Delta$W[l] $+\lambda$W[l])

which is the improved version of the gradient decent step for a batch, in order for the training to benefit from the additional part in J.

The average cost function for one epoch in equation (4.38) is updated to

$$\hat{J}=\frac{1}{m}\sum_{tp=1}^{m}J=\frac{1}{2m}\sum_{tp=1}^{m}\left(\left\|\mathbf{y}-\mathbf{h}^{(n)}\right\|^2+\lambda\sum_{i,j}\left(w_{ij}^{(l)}\right)^2\right) \tag{4.44}$$

In the outline of the implementation of the FNN, the line

avg_cost = avg_cost + equation (4.11)

is replaced with the line

avg_cost = avg_cost + equation (4.39)

4.2.6 A test of the FNN

The Python machine learning library Scikit-Learn provides a version of the MNIST data set which contains 1797 digital images of hand written numbers between 0 and 9. Each image contains 8 x 8 = 64 pixels. The images are organized as data pairs with 1797 data examples in the (1797 x 64) matrix $\mathbf{X}_{\text{data}}$ and with the corresponding 1797 targets in the vector (array in Python) $\mathbf{y}_{\text{data}}$. As already explained, the data set $(\mathbf{X}_{\text{data}}, \mathbf{y}_{\text{data}})$ is splitted into a training set $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ with training pairs and a test set $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ with test pairs. In order to use the targets $\mathbf{y}_{\text{train}}$ in the FNN, each target y, which is just a single integer between 0 and 9, has to be transformed to its corresponding *one hot* vector representation $\mathbf{y}$ with length 10, which will be the number of nodes in the output layer. For example is $y = 8$ transformed to $\mathbf{y} = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ and $y = 4$ is transformed to $\mathbf{y} = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$. Since the number of pixels in the images are 64, the input layer has 64 nodes.

The data example in figure (4.4) shows the black and white digital image of a hand written number 0. The (8 x 8) matrix, which is shown in equation (4.45) and converted from a (1 x 64) vector (array), shows that the corresponding pixel values are integers between 0 and 15. Each integer denotes the black and white intensity of the pixels between black (integer = 0) and white (integer = 15).

The test set contains 30 percent of the images, which is 540 images, and the training set contains 70 percent of the images, which is 1257 images. With a bit of experimenting, the following hyper-parameters were chosen:

- One hidden layer with 25 nodes
- The scale parameter in the sigmoid function $s = 1$
- The gradient decent step size $\alpha = 0.1$
- The regularization parameter $\lambda = 0.0001$

- The batch size $bs = 5$

This is probably not the most optimal choice of hyper-parameters, but this is just a test of the FNN and the result is actually not bad at all. After 200 epochs, the test accuracy is 96 percent. Figure (4.5) shows the average cost function for each epoch and figure (4.6) shows the test accuracy in percent after each epoch. The FNN learns quite fast to classify the digital images of the hand written numbers, and most of the learning occurs during the first 50 epochs.

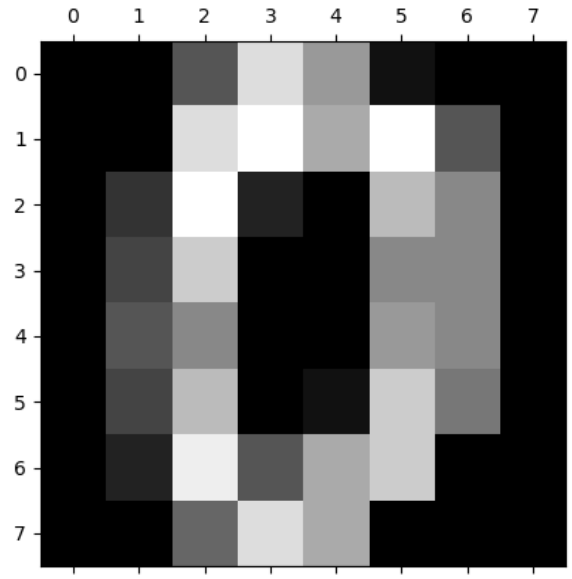

Figure 4.4: A 64 pixels digital image of a hand written number 0.

$$\begin{bmatrix} 0. & 0. & 5. & 13. & 9. & 1. & 0. & 0. \\ 0. & 0. & 13. & 15. & 10. & 15. & 5. & 0. \\ 0. & 3. & 15. & 2. & 0. & 11. & 8. & 0. \\ 0. & 4. & 12. & 0. & 0. & 8. & 8. & 0. \\ 0. & 5. & 8. & 0. & 0. & 9. & 8. & 0. \\ 0. & 4. & 11. & 0. & 1. & 12. & 7. & 0. \\ 0. & 2. & 14. & 5. & 10. & 12. & 0. & 0. \\ 0. & 0. & 6. & 13. & 10. & 0. & 0. & 0. \end{bmatrix} \tag{4.45}$$

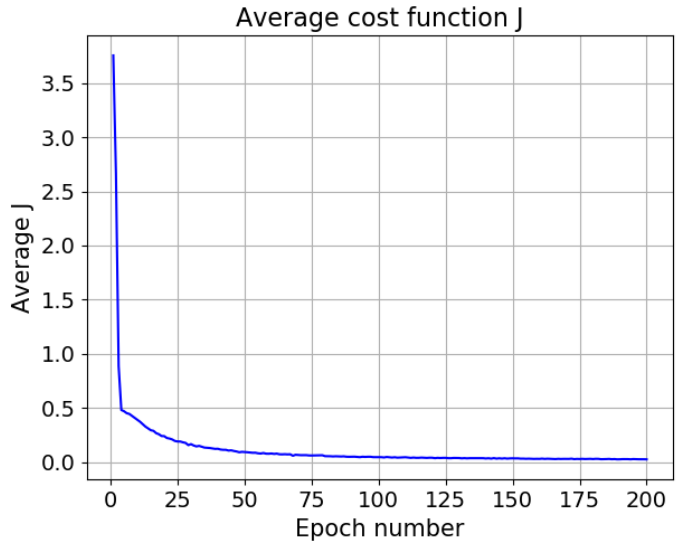

Figure 4.5: The average cost function for each epoch for the MNIST data set.

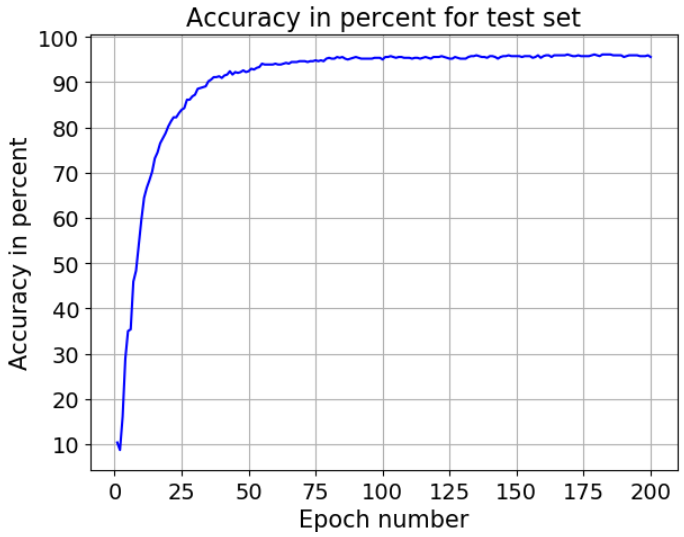

Figure 4.6: The test accuracy after each epoch for the MNIST data set.

4.3 A FNN using Tensorflow 2 with the Keras API

Now that the basic understanding of how a FNN works are in place, a FNN can be implemented by using the Tensorflow 2 with the Keras API. The technicalities in the implementation of the FNN in section 4.2 will now take place in the background, and the concepts like layers, initialization of weights, activation function, cost (or loss) function, gradient decent method and regularization can be seen as tools in the toolbox of Tensorflow 2 and Keras. These tools can be found through a search in Google or another search engine. For example if we want to read about a particular layer in Tensorflow 2 and Keras, searching for *Tensorflow 2 + layers* will suggest the following link, see [21], which gives the links to the layers in Tensorflow 2 and Keras, for example the *Dense* layer which is used in FNNs.

4.3.1 The Dense layer

The hidden layer(s) and the output layer used in the FNN are called *Dense*. In order to define the structure of a Dense layer, it can access the necessary tools in the Tensorflow 2 and Keras toolbox through arguments. The choices of some of these arguments are presented here.

Layer size
The size of each Dense layer is defined by the number of nodes in the layer, or by the *dimensionality of the output space* as Tensorflow 2 and Keras put it. The dimensionality of the output space is the dimensionality to the output $\mathbf{h}$ from the activation function applied to the layer, see equation (4.7). The dimensionality of the input space is defined by the output dimensionality of the former layer in the FNN. For the first Dense layer, the input dimensionality is defined by the length of the array with the input node values. The input dimensionality and the output dimensionality defines the dimensions of the layer's weight matrix $\mathbf{W}$.

The ReLU activation function
The following problem and the solution of it is described in chapter 4 in the eBook *Coding the Deep Learning Revolution*, which can be purchased at [18]. The sigmoid activation function $f(z)$ and the derivative $f'(z)$ of the function are introduced in equation (4.9) and (4.10) respectively, and both the function itself and its derivative are shown in figure (4.2). Outside a given interval around $z = 0$, the value of $f'(z)$ becomes quite small and it approaches zero as $|z|$ increases. Too small values of $f'(z)$ can cause

problems in the updating of the weights $w_{ij}^{(l)}$ in $\mathbf{W}$ and the bias weights $b_i^{(l)}$ in $\mathbf{b}$ with the gradient decent method in equation (4.13) and (4.14) respectively, because it can lead to values of $\partial J/\partial \mathbf{W}$ and $\partial J/\partial \mathbf{b}$ which are too small in order to give an update of $\mathbf{W}$ and $\mathbf{b}$ of any significance. The result is a poor training outcome for the FNN, or even a complete stalling of it.

In order to show what happens with $\partial J/\partial \mathbf{W}$, we can take a look at a FNN with two hidden layers, like the one in figure (4.1). For the output layer, equation (4.36) and equation (4.34) give:

$$\frac{\partial J}{\partial \mathbf{W}^{(3)}} = \boldsymbol{\delta}^{(4)} \left(\mathbf{h}^{(3)}\right)^T = -\left(\mathbf{y} - \mathbf{h}^{(4)}\right) \circ f'\left(\mathbf{z}^{(4)}\right) \tag{4.46}$$

For the two inner layers, equation (4.36) and the back propagation method in equation (4.35) give:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}^{(2)}} &= \boldsymbol{\delta}^{(3)} \left(\mathbf{h}^{(2)}\right)^T \\ &= \left[\left((\mathbf{W}^{(3)})^T \boldsymbol{\delta}^{(4)}\right) \circ f'\left(\mathbf{z}^{(3)}\right)\right] \left(\mathbf{h}^{(2)}\right)^T \end{aligned} \tag{4.47}$$

and

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}^{(1)}} &= \boldsymbol{\delta}^{(2)} \left(\mathbf{h}^{(1)}\right)^T = \boldsymbol{\delta}^{(2)} (\mathbf{x})^T \\ &= \left[\left((\mathbf{W}^{(2)})^T \boldsymbol{\delta}^{(3)}\right) \circ f'\left(\mathbf{z}^{(2)}\right)\right] (\mathbf{x})^T \end{aligned} \tag{4.48}$$

These three equations (equation (4.46), (4.47) and (4.48)) show that the number of $f'(z)$ terms increases by one for each step backwards through the layers: $\partial J/\partial \mathbf{W}^{(3)}$ is proportional to $f'\left(\mathbf{z}^{(4)}\right)$, which is expressed as

$$\frac{\partial J}{\partial \mathbf{W}^{(3)}} \propto f'\left(\mathbf{z}^{(4)}\right) \tag{4.49}$$

and further we have

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} \propto f'\left(\mathbf{z}^{(4)}\right) \circ f'\left(\mathbf{z}^{(3)}\right) \tag{4.50}$$

and

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} \propto f'\left(\mathbf{z}^{(4)}\right) \circ f'\left(\mathbf{z}^{(3)}\right) \circ f'\left(\mathbf{z}^{(2)}\right) \tag{4.51}$$

So the problem with poor training outcome will increase with an increasing number of hidden layers because an increasing number of $f'(\mathbf{z})$, which quite possibly has small values, will be multiplied with each other. The problem is called *The vanishing gradient problem*. It can be described in the same way for the bias weights in $\mathbf{b}^{(l)}$. If poor training outcome occurs due to the vanishing gradient problem, the ReLU (Rectify Linear Unit) function can be a better option as the activation function. It is defined as

$$f(z) = max(0, z) \tag{4.52}$$

The function solves the vanishing gradient problem for $z > 0$, but if $z < 0$, weights and bias weights will be excluded from the feed forward pass, and the update of weights with the gradient decent method will stop. In Tensorflow2 and Keras, a hyper-parameter a can be included so that the ReLU activation function becomes

$$f(z) = max(az, z) \tag{4.53}$$

which solves the problems if $z < 0$. Figure (4.7) shows the function and its derivative for $a = 0$ and $a = 0.1$. In the implementation of a FNN with Tensorflow2 and Keras, the ReLU activation function is applied to the hidden layer(s) instead of the sigmoid activation function.

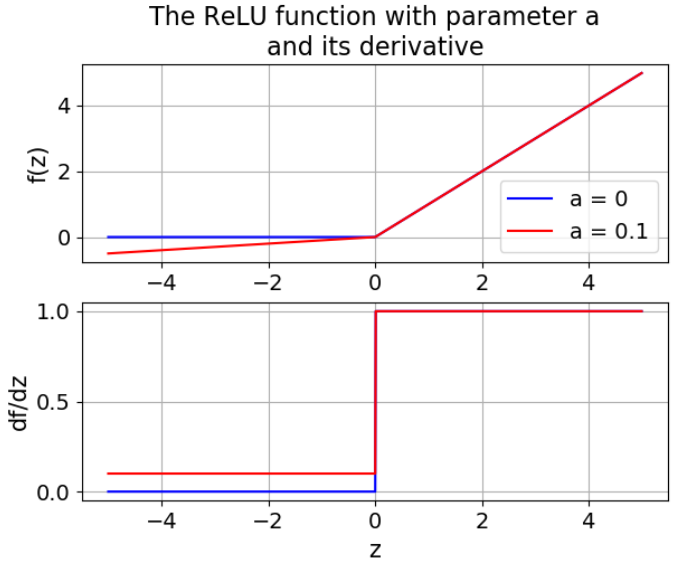

Figure 4.7: The ReLU function with parameter $a = 0$, and $a = 0.1$ and its derivative.

The softmax activation function

An activation function called softmax is applied to the output layer. It is well explained in [22]. The result of applying the softmax activation function is output node values with sum equal to one, where each output node value represent the probability of the correct output or target for that particular node. For a network with three output nodes, like the one in figure (4.1), and with n layers, equation (4.7) can as an example give the values

$$\mathbf{z}^{(n)} = \mathbf{W}^{(n-1)}\mathbf{h}^{(n-1)} + \mathbf{b}^{(n-1)} = \begin{bmatrix} 1.0 \\ 1.6 \\ 1.1 \end{bmatrix}$$

For this example the softmax function becomes

$$h_i^{(n)} = f\left(z_i^{(n)}\right) = \frac{e^{z_i^{(n)}}}{\sum_{j=1}^{3} e^{z_j^{(n)}}}$$

which gives

$$h_1^{(n)} = f\left(z_1^{(n)}\right) = \frac{e^{z_1^{(n)}}}{e^{z_1^{(n)}} + e^{z_2^{(n)}} + e^{z_3^{(n)}}} = \frac{e^{1.0}}{e^{1.0} + e^{1.6} + e^{1.1}} = 0.2546$$

$$h_2^{(n)} = f\left(z_2^{(n)}\right) = \frac{e^{z_2^{(n)}}}{e^{z_1^{(n)}} + e^{z_2^{(n)}} + e^{z_3^{(n)}}} = \frac{e^{1.6}}{e^{1.0} + e^{1.6} + e^{1.1}} = 0.4640$$

$$h_3^{(n)} = f\left(z_3^{(n)}\right) = \frac{e^{z_3^{(n)}}}{e^{z_1^{(n)}} + e^{z_2^{(n)}} + e^{z_3^{(n)}}} = \frac{e^{1.1}}{e^{1.0} + e^{1.6} + e^{1.1}} = 0.2814$$

or

$$\mathbf{h}^{(n)} = f\left(\mathbf{z}^{(n)}\right) = \begin{bmatrix} 0.2546 \\ 0.4640 \\ 0.2814 \end{bmatrix}$$

Initialization of weights

The following problem and the solution of it is described in chapter 5 in the eBook *Coding the Deep Learning Revolution*, which can be purchased at [18]. In the implementation of a FNN without the support of Tensorflow 2 and Keras, the weights $w_{ij}^{(l)}$ in the weight matrices $\mathbf{W}^{(l)}$ and the bias weights $b_i^{(l)}$ in the bias weight vectors $\mathbf{b}^{(l)}$ were initialized with random values uniformly distributed in the interval $[0, 1)$. This is a simple approach, but not the most

optimal one for initialization of the weights and bias weights because it can eventually lead the gradient decent method to approach a local minimum in the cost function which leads to a poor training outcome. The reason is that the variance of $z_i^{(l)}$ in equation (4.8) eventually can become large during the training of the network. The result is a large span in the values of $z_i^{(l)}$, which will lead to several small values of $f(z_i^{(l)})$, at least for the sigmoid activation function in equation (4.9) which is shown in figure (4.2). This is because too small values of $f'(z)$ can cause problems in the update process of the weights $w_{ij}^{(l)}$ in the weight matrices $\mathbf{W}^{(l)}$ and the bias weights $b_i^{(l)}$ in the bias weight vectors $\mathbf{b}^{(l)}$ in the gradient decent method in equation (4.13) and (4.14) respectively, because it can lead to too values of $\partial J/\partial \mathbf{W}^{(l)}$ and $\partial J/\partial \mathbf{b}^{(l)}$ which is too small in order to give an update of $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ of any significance. The result is poor training of the network or even a complete stalling of it.

A solution to this problem is given in [23]. The weight initialization described in this article is based on the idea that optimal training of the network occurs if the initialization of the weights can keep the variance of $z_i^{(l)}$ constant through the layers. In Tensorflow 2 and Keras this method for the initialization of the weights is called Glorot Normal and Glorot Uniform after Xavier Glorot, who is one of the authors of [23]. The initialization of the weights described in [24] is based on the same ideas about the variance, and might work better than the initialization described in [23] when the ReLU activation function is used, because $f'(z) = 0$ when $z \leq 0$, see equation (4.53) and figure (4.7) for $a = 0$. In Tensorflow 2 and Keras this method for the initialization of the weights is called He Normal and He Uniform after Kaiming He, who is one of the authors of [24].

Another method for the initialization of the weights is the Orthogonal method. With this method each weight matrix is initialized as an orthogonal matrix. The Tensorflow 2 manual briefly describes how an orthogonal weight matrix is obtained, see [25]. The benefits of the initialization of weights with the Orthogonal method is investigated in [26]

L2 regularization

What it means that a network is over-fitted to the training data and the method of L2 regularization as a way to avoid over-fitting, was covered in section 4.2.5. In Tensorflow 2 and Keras, L2 regularization is accessed through an argument in the definition of the Dense layer structure.

4.3.2 The Dropout layer

In order to avoid over-fitting, regularization through a layer called a *Dropout* layer can be used in addition to L2 regularization. Dropout regularization is well explained in [28] and was introduced in [29].

In a Dropout layer in Tensorflow 2 and Keras, a dropout rate is chosen. A dropout rate of for example 0.2 means that 20 percent of the input node values $h_i^{(l)}$ (or x_i for $l = 1$) for the next layer will be randomly chosen and excluded from the feed forward pass, and the weights $w_{ij}^{(l)}$ connected to the excluded nodes will not be updated in the updating process when going backwards through the layers. The result is that the weights become less dependent of each other and the network itself becomes less dependent of particular weights. The idea is that more independent weights will lead to a network which is less sensitive to insignificant details and arbitrary noise in the training set and a network which is better at finding the general patterns in the set. A Dropout layer is only active during training and not during the testing of the network.

4.3.3 The cost (or loss) function

In section 4.2.1 a simple cost (or loss) function, see equation (4.11), which only calculated the L^2 norm of the difference between the output $\mathbf{h}^{(n)}$ of a feed forward pass and the *one hot* vector representation $\mathbf{y}$ of the target y, was introduced. When Tensorflow 2 and Keras are used, this cost function is replaced with a cost function called *CategoricalCrossentropy*, or with the cost function called the *SparseCategoricalCrossentropy*. More about the difference between them in a moment. According to chapter 1 in the eBook *Coding the Deep Learning Revolution*, which can be purchased at [18], the cross entropy cost function is a cost function which is commonly used in the task of classifying the data examples of different categories. An explanation of the theory behind it can be found in [27].

When using the CategoricalCrossentropy cost function in Tensorflow 2 and Keras, each target y in $\mathbf{y}_{train}$ needs to be transformed to its corresponding one hot vector representation $\mathbf{y}$, like it was shown in the conjunction with the cost function in equation (4.11). Using the SparseCategoricalCrossentropy cost function instead, the one hot transformation of the targets y is not necessary. In the implementation of a FNN with Tensorflow 2 and Keras, the SparseCategoricalCrossentropy cost function will be used.

4.3.4 Update of the weights with the Adam optimizer

In section 4.2.2, gradient decent with back propagation was introduced as a method for updating the weights $w_{ij}^{(l)}$ and the bias weights $b_i^{(l)}$. When Tensorflow 2 and Keras are used, an algorithm called the *Adam optimizer* will be used in order to update the weights and bias weights. The Adam optimizer is used in the examples in the eBook *Coding the Deep Learning Revolution*, which can be purchased at [18]. The Adam optimizer was introduced in [30] and a summary about how the optimizer works can be found in [31]. A hyper-parameter called the step size α was a part of the gradient decent method in section 4.2.2. α is also a part of the Adam optimizer and is referred to as the learning rate. The optimizer also includes three other parameters which is β_1, β_2 and ϵ. For these three parameters, the default values in Tensorflow 2 and Keras are used, but the α parameter will be tested for a few other values in the process of optimizing the training of the network.

4.3.5 An outline of a FNN using Tensorflow 2 and Keras

In the same way as described in section 4.2.4, the Scikit-Learn function *train_test_split* and the Scikit-Learn functions *fit_transform* and *transform* are used in order to prepare the data set for the FNN. The outline of the FNN using Tensorflow 2 and Keras is based on the implementation of the FNN in [19].

#Comment: A sequencial model in Tensorflow 2 and Keras is just a network
#with layers stacked after each other, like the network in figure (4.1) is an
#illustration of.
model = Sequential()

#Comment: The input layer with the dimension of the data examples
model.add(Input(shape = number of input nodes))

#Comment: A Dense layer with the following arguments:
#1) The number of nodes in the layer.
#2) The activation function.
#3) Method for initialization of the weights w_{ij}.
#4) Method for initialization of the bias weights b_i.
#5) L2 regularization considering the weights.
#6) L2 regularization considering the bias weights.
model.add(Dense(Argument 1, $\cdots$, Argument 6))

#Comment: A Dropout layer. The dropout rate is 0.1.
model.add(Dropout(0.1))

#Comment: The output Dense layer with the softmax
#activation function.
model.add(Dense(number of nodes, activation=softmax))

#Comment: The configuration of the training and testing of the
#model includes these three arguments:
#1) The Adam optimizer with the learning rate
(or step size) α.
#2) The SparseCategoricalCrossentropy cost function.
#3) The metrics to be performed is the calculation of the
# accuracy of the model. The accuracy is just the
# frequency of correct classifications, which is a number
# between 0 to 1.
model.compile(Argument 1, Argument 2, metrics=['accuracy'])

#Comment: The training and the testing of the model, including
#the arguments, which are self-explanatory.
history = model.fit($\mathbf{X}_{train}$, $\mathbf{y}_{train}$, batch size, number of epochs,
  validation_data = ($\mathbf{X}_{test}$, $\mathbf{y}_{test}$))

#Comment: *history* is an object with the following attributes:
#1)A record of the training cost (loss) values for all the epochs.
#2)A record of the training accuracy for all the epochs.
#3)A record of the testing cost (loss) values for all the epochs.
#4)A record of the testing accuracy for all the epochs.

4.3.6 A test of the FNN

As in section 4.2.6, the MNIST data set from the Scikit Learn library with 1797 64 pixels digital images of hand written numbers between 0 and 9, is used to test the Tensorflow and Keras version of a FNN. The size of the test set and the training set is unchanged, which means a test set with 30 percent of the images, which are 540 images, and a training set with 70 percent of the images, which are 1257 images. The method used for initialization of weights and bias weights is He Normal. With a bit of experimenting, the following hyper-parameters were chosen:

- One hidden layer with 64 nodes.

- The learning rate (or step size) $\alpha = 0.0001$.
- The regularization parameter $\lambda = 0.001$.
- The ReLU activation function with parameter $a = 0$.
- The batch size $bs = 3$.
- A separate Dropout layer prior to the hidden layer and to the output layer with dropout rate = 0.

After 60 epochs, the test accuracy is 98 percent. Figure (4.8) shows the average cost function for each epoch and figure (4.9) shows the test accuracy in percent after each epoch.

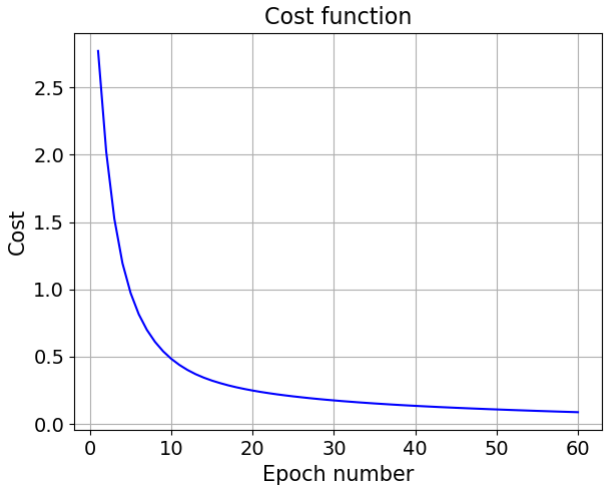

Figure 4.8: The average cost function for each epoch for the MNIST data set.

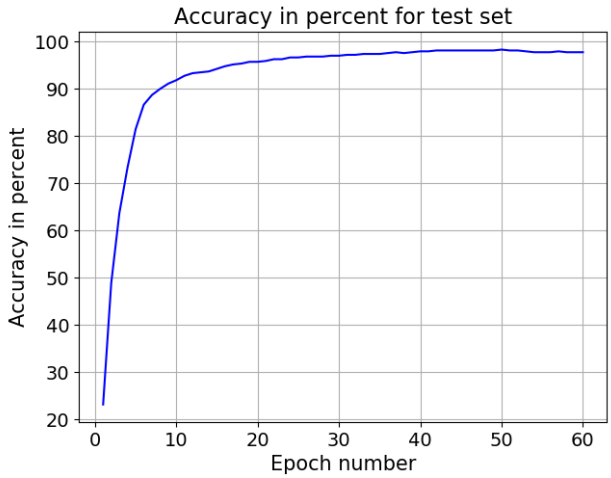

Figure 4.9: The test accuracy after each epoch for the MNIST data set.

4.4 A RNN with LSTM cells using Tensorflow 2 and Keras

The main difference between a Feed-forward Neural Network (FNN) and a Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells was explained in section 4.1. A FNN can be used classify the data examples in a data set where the data example are static, which means that the order of the features in the data set doesn't matter. A example is a data set which contains digital images with a given number of pixels and where the images can be classified in a given number of categories. On the other hand, if it is a time dependence between the features in the data set, the set contains dynamic data examples. A FNN will not be able to discover these time dependencies and use it in order to improve the training outcome in the classification task.

In order to utilize eventual time dependencies between the measurements in the impedance spectra and thereby possibly improve the training outcome of the tissue classification compared with the FNN results, a RNN with LSTM cells is also implemented by using Tensorflow 2 and Keras. Both RNNs and LSTM cells are described in chapter 10 in the eBook *Coding the Deep Learning Revolution*, which can be purchased at [18]. A good description of RNNs and LSTM cells are also found in [32] LSTM cells was first introduced

in [33].

In a RNN, the data elements in the data examples of the data set enter the RNN in steps which are called time steps or recurrent steps. If we want to classify the 8 x 8 = 64 pixels images of hand written numbers between 0 and 9 from the MNIST data set, which was introduced in section 4.2.6, the data examples can enter the network from one input node in 64 time steps from a 64 x 1 vector with the pixels. The data examples can also be organized as for example a 8 x 8 matrix instead of an vector, which enters the network one row at the time from 8 input nodes in 8 time steps. In that way the training process of the network will go faster, and since there is no time dependency between the pixels in an image, it will not have any negative effect on the result of the training. The impedance spectra contain the real part Z_x and the imaginary part Z_y of 50 measurement points on a logarithmic scale between 10 Hz and 1 MHz. (From Z_x and Z_y, the modulus $|Z|$ and the phase angle θ can be calculated, see chapter 2.) In order to help the network in the training process, the discrete values of the derivatives of Z_x and Z_y can be included, but for simplicity they are not part of the explanation here. The data examples in the data set then contain 50 Z_x values and 50 Z_y values. They can be organized as 100 x 1 vector with the 50 Z_x values followed by the 50 Z_y values, which enter the network one element at the time from one input node in 100 time steps. They can also be organized as a 50 x 2 matrix with the 50 Z_x values in the first column and with the 50 Z_y values in the second column, which enter the network in one row at the time from two input nodes in 50 time steps. In that way the training process of the network will go faster. Since there is no physical connection between the last Z_x value and the first Z_y value, using two input nodes seem to be the best choice. The modulus $|Z|$ and the phase angle θ can also be used instead of Z_x and Z_y.

Figure (4.10) is an illustration of a RNN with 2 nodes in the input layer (layer 1) and two nodes in the first hidden layer (layer 2). In the figure $X1,t = x_1^{(t)}$ and $X2,t = x_2^{(t)}$ are the input values to the network at time step t. $h1,t = h_1^{(l=2,t)}$ is the output from node 1 in the hidden layer (layer 2: l = 2) at time step t and $h2,t = h_2^{(l=2,t)}$ is the output from node 2. The figure illustrates that the output values from the nodes in the hidden layer returns back again as an input to the node it left together with the input values from the next time step t+1. In this way the hidden layer keeps a memory of previous time steps when a new time step with input values

enter the network. The return of the output values from the nodes back to the nodes again is the reason why the network is called a *Recurrent Neural Network* (RNN).

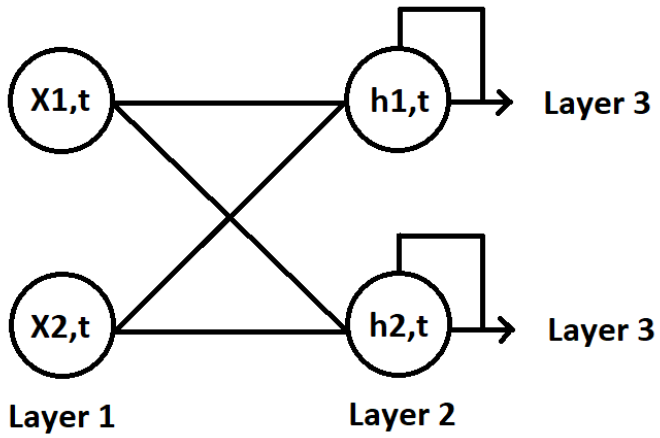

Figure 4.10: The input layer and the first hidden layer in a RNN network.

A less detailed illustration of the hidden layer in figure (4.10) during n time steps is shown in figure (4.11). In the figure

$$X, t = \mathbf{x}^{(t)} = \left[x_1^{(t)}, x_2^{(t)}\right]$$

and

$$h, t = \mathbf{h}^{(l=2,t)} = \left[h_1^{(l=2,t)}, h_2^{(l=2,t)}\right]$$

The layer has one output to the next layer for all the time steps, which also are called recurrent steps due to the return of the output node values back into the layer. It is called a *many to many* configuration, and requires that the next layer also is a layer which is designed to handle time steps (or recurrent steps).

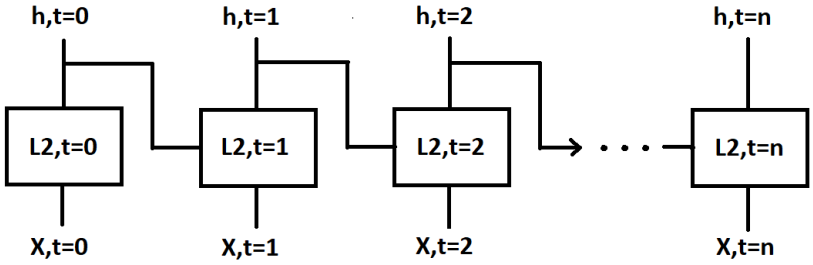

Figure 4.11: The hidden layer during n time steps (recurrent steps) and with a many to many configuration.

If the next layer is the output layer with the softmax activation function, which is a *Dense* layer described in 4.3.1, the inner RNN layer can only have an output from the last time step. This configuration is called a *many to one* configuration, and it is shown in figure (4.12)

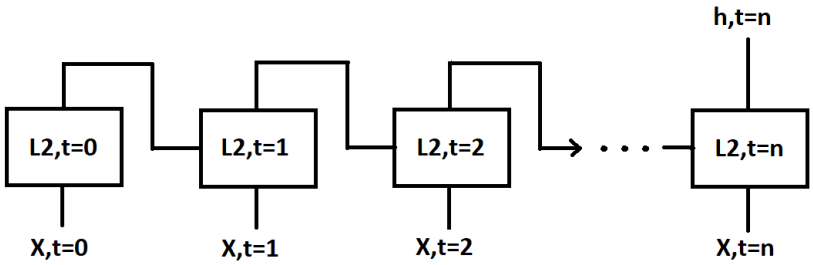

Figure 4.12: The hidden layer during n time steps (recurrent steps) and with a many to one configuration.

When the node values in $\mathbf{h}$ pass through from one layer to the next layer in a FNN during the feed forward pass, we have seen that a weight matrix $\mathbf{W}$, a bias weight vector $\mathbf{b}$ and an activation function f are involved in the transformation of them. A similar process involving weights and an activation function takes place in a layer during the transformation of the node values in $\mathbf{h}$ from one time step to the next in a RNN with classical/ordinary layers (not LSTM cells). The process of updating the weights involved during the transformation of $\mathbf{h}$ from one time step to the next, involves the

gradient decent method which was presented in section 4.2.2. In the updating process of the weights, several time steps will then involve several derivatives of the activation function multiplied with each other. A similar process was described in the ReLU activation function part in section 4.3.1. The updating process might therefore lead to the vanishing gradient problem (also described in the ReLU activation function part in section 4.3.1). The solution is not the ReLU activation function this time, but the LSTM cell.

An explanation of how a LSTM cell works, apart from stating that it solves the vanishing gradient problem, will be too comprehensive here. Schematically it looks quite similar to the hidden layer sketched as rectangles for several time steps in figure (4.11) and figure (4.12). In addition the LSTM cell also contains a so called *Cell State* $\mathbf{C}^{(l,t)}$, which together with the node values $\mathbf{h}^{(l,t)}$ is updated from one step to the next one. The internal structure of the LSTM cell is more complex than in a classical/ordinary RNN layer. It has three so called gates called *the forget gate layer*, *the input gate layer* and *the output gate layer* with their own weights and activation functions. Due to these gates in the LSTM cells, the RNN can learn which information to keep and which information to forget from previous steps. The vanishing gradient problem is avoided and a better training outcome for the RNN is achieved. In Tensorflow 2 and Keras a LSTM cell with its own internal structure with gates or gate layers, is called a layer: a LSTM layer.

Using LSTM layers (cells) in a RNN in Tensorflow 2 and Keras is not very different from using the Dense layers described in section 4.3.1 in a FNN. In the same way as for a Dense layer, a LSTM layer can access the necessary tools in the Tensorflow 2 and Keras toolbox through arguments. The number of nodes in the layer, type of activation function, the method for the initialization of weights and for the L2 regularization method are arguments already described for the Dense layer, and which are used together with a LSTM layer too. In addition the LSTM layer also has similar arguments for the time steps (or recurrent steps). It has an argument for the recurrent activation function, for the method for the initialization of the recurrent weights and for the recurrent L2 regularization.

For the activation function and the recurrent activation function the default functions are used, which are the hyperbolic tangent function and the sigmoid function respectively. Dropout, which was introduced in section 4.3.2 can also be used together with a LSTM layer. Instead of using a separate

Dropout layer, an argument in the LSTM layer for the dropout rate and the recurrent dropout rate respectively can be used.

The output layer will be a Dense layer with the softmax activation function introduced in section 4.3.1. A Dense layer is not designed for time steps, so the hidden LSTM layer before the output layer must have a many to one configuration, see figure (4.12). The LSTM argument *return_sequences = False* defines the layer as a many to one layer. The LSTM layer argument *return_sequences = True* defines the layer as a many to many layer, see figure (4.11). This configuration is used if the next layer is a LSTM layer also.

The SparseCategoricalCrossentropy cost function introduced in section 4.3.3 and the Adam optimizer introduced in section 4.3.4 are also used together with LSTM layers.

4.4.1 An outline of a RNN with a LSTM layer (cell) using Tensorflow 2 and Keras

In the same way as described in section 4.2.4, the Scikit-Learn function *train_test_split* and the Scikit-Learn functions *fit_transform* and *transform* are used in order to prepare the data set for the RNN with LSTM layers. Further, the outline is quite similar to the outline of a FNN using Tensorflow 2 and Keras in section 4.3.5, and it is based on the implementation of the RNN with LSTM layers in [19].

#Comment: A sequencial model in Tensorflow 2 and Keras is just a network
#with layers stacked after each other, like the network in figure (4.1) is an
#illustration of.
model = Sequential()

#Comment: The input layer with the dimension of the data examples
inp = (number of time steps, number of elements in each time step)
model.add(Input(shape = inp))

#Comment: A LSTM layer with the following arguments:
#1) The number of nodes in the layer.
#2) Method for initialization of the weights.
#3) Method for initialization of the recurrent weights.
#4) L2 regularization considering the weights.
#5) L2 regularization considering the recurrent weights.

```
#6) Dropout rate considering the weights.
#7) Dropout rate considering the recurrent weights.
#8) Dependent on whether the next layer is a LSTM layer or
#    a Dense layer with the softmax ativation function:
#    return_sequences = True (LSTM layer next) or
#    return_sequences = False (Dense layer next).
model.add(LSTM(Argument 1, ···, Argument 8))


#Comment: The output Dense layer with the softmax
#activation function.
model.add(Dense(number of nodes, activation=softmax))

#Comment: The configuration of the training and testing of the
#model. The three arguments are explained in in section 4.3.5.
model.compile(Argument 1, Argument 2, metrics=['accuracy'])

#Comment: The training and the testing of the model, including
#the arguments, which are self-explanatory. The history object
#is explained in section 4.3.5.
history = model.fit(X_train, y_train, batch size, number of epochs,
          validation_data = (X_test, y_test))
```

4.4.2 A test of the RNN with one LSTM layer (LSTM cell)

As in section 4.2.6, the MNIST data set from the Scikit Learn library with 1797 64 pixels digital images of hand written numbers between 0 and 9, is used to test a RNN with a LSTM layer. The size of the test set and the training set is unchanged, which means a test set with 30 percent of the images, which are 540 images, and a training set with 70 percent of the images, which are 1257 images. Each image is organized as a 8 x 8 matrix, which enters the network one row at the time in 8 time steps or recurrent steps. The method used for initialization of weights and recurrent weights is Glorot Normal. With a bit of experimenting, the following hyper-parameters were chosen:

- One hidden LSTM layer with 64 nodes.
- The learning rate (or step size) $\alpha = 0.0001$.
- The regularization parameter and recurrent regularization parameter $\lambda = 0.001$.

- The batch size $bs = 3$.
- In the LSTM layer, dropout rate = 0 and recurrent dropout rate = 0.

After 60 epochs, the test accuracy is 96 percent. Figure (4.13) shows the average cost function for each epoch and figure (4.14) shows the test accuracy in percent after each epoch.

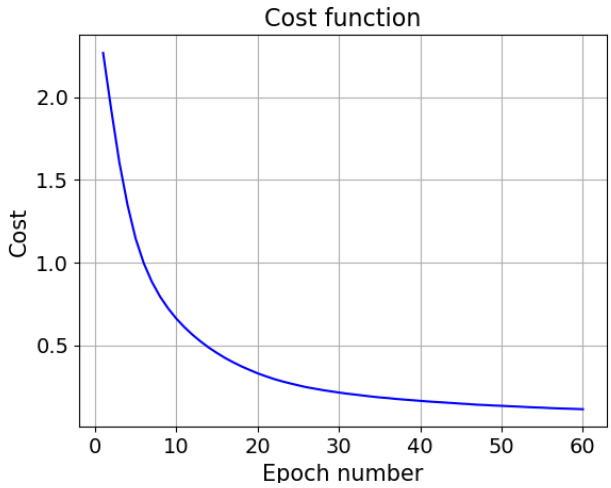

Figure 4.13: The average cost function for each epoch for the MNIST data set.

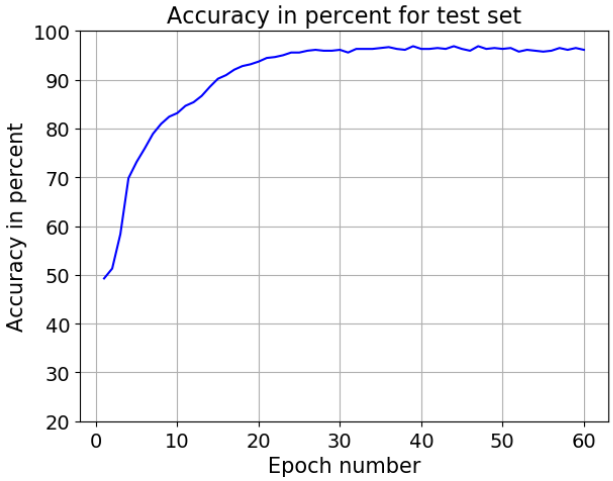

Figure 4.14: The test accuracy after each epoch for the MNIST data set.

Chapter 5

Impedance measurements and data sets

5.1 Introduction

This chapter describes the electrical impedance measurements in fat and muscle tissue in a newly deceased pig during two separate afternoons/evenings at Rikshospitalet in Oslo. The chapter also describes the two data sets with the established impedance spectra, one for each of the two needle types used to measure the tissue impedance, which will be the data material in the analyzes with the Artificial Neural Networks (ANNs) described in chapter 4.

5.2 The impedance measurements

The impedance measurements in fat and muscle tissue in a newly deceased pig were performed during two separate afternoons/evenings at Rikshospitalet in Oslo under the guidance of and in collaboration with my co-supervisor Håvard Kalvøy. The measurements were performed after other researches had completed their research work on the pig when it was under anesthesia, and the pig was put to death. At the first afternoon/evening, a Stimuplex A needle was used in order to perform the measurements. At the second afternoon/evening, a SonoPlex needle was used. The needles were new, fresh from their sealed packaging. The two needle types are introduced in chapter 3. The two afternoons/evenings were organized in the following way

- The time of death of the pig was noted. The impedance measurements

in the fat and muscle tissue started approximately 2 hours after the time of death.

- The Zurich impedance analyzer and the laptop with the instrument's software called LabOne was prepared for the impedance measurements. The two Sandberg Powerbank 20000 for Laptops were used as the power source for the Zurich instrument. Each powerbank lasted just over 2 hours before it had to be replaced with the other powerbank, and recarged. Figure (5.1) shows a picture of the Zurich instrument, the powerbank and the laptop with the LabOne software during the impedance measurements.

- The preconditioning of the needle followed the same procedure as in chapter 3. The needle was preconditioned in saline. Several impedance spectra in a row between 10 Hz and 1 MHz were established until the impedance spectra became stable. The needle was filled with saline prior to the preconditioning, and blown with air inside afterwards in order to remove the saline inside the needles.

- The Ambu BlueSensor Q Ag/AgCl ECG electrode was introduced in chapter 2. Two ECG electrodes were attached to the skin of the pig and connected to the Zurich instrument's Counter port and Reference port respectively. The electrodes was attached relatively close to each other.

- My co-supervisor made the incisions in the pig in order to access fat and muscle tissue for the impedance measurements. Impedance measurements was not performed in the subdermal fat just below the skin of the pig. Figure (5.1) shows a picture of the pig with the two ECG electrodes attached to the skin, and the Stimuplex A needle positioned in the tissue exposed by one of the incisions.

- Since the pig was dead during the impedance measurements, the tissue temperature dropped gradually. The drop in temperature could possibly affect the impedance spectra. The measurements started in the muscle tissue. In order to minimize the possible effect of the temperature drop on the impedance spectra in the data set, only 20 impedance spectra at the time were established in one tissue type, before the impedance measurements started in the other tissue type. Between the impedance measurements in the two tissue types, the needle was rinsed and flushed inside with saline. The needle was blown with air

inside afterwards in order to remove the remaining saline. Five sets with 20 spectra were established in each of the two tissue types.

- The parameters specified in LabOne were unchanged from the tests with the Zurich instrument described in chapter 3, so the time it took to establish an impedance spectrum was still 17 seconds. The parameters are repeated a little later in this chapter for the sake of order. In each of the two tissue types, the position of the needle was changed between the establishment of each impedance spectrum. Sometimes the position was changed just a little by pushing the needle 1 mm to 2 mm deeper into the tissue. Other times the position of the needle was changed to another place in the incision or to the tissue in another incision. Depending on the time used to find new positions for the needle, establishing the 20 impedance spectra in one of the two tissue types took between 10 minutes and 20 minutes.

- After the measurements were completed, the hospital's procedure for animals sacrificed for medical research purposes were followed: My co-supervisor closed the incisions. Then the pig was wrapped in two large paper bags and transported to the cold store for freezing.

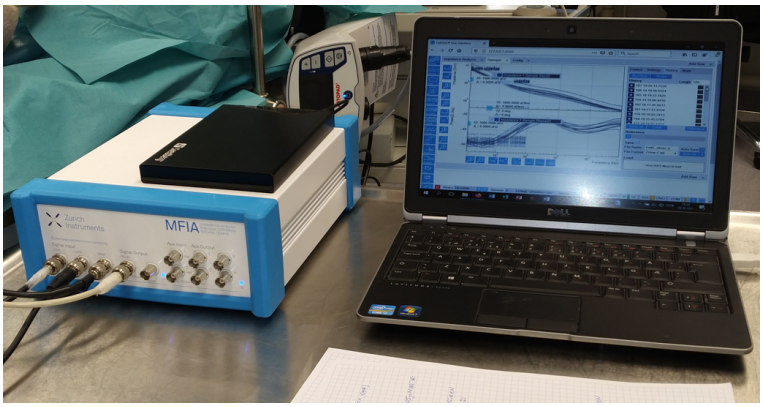

Figure 5.1: The Zurich instrument, the powerbank and the laptop with the LabOne software during the impedance measurements.

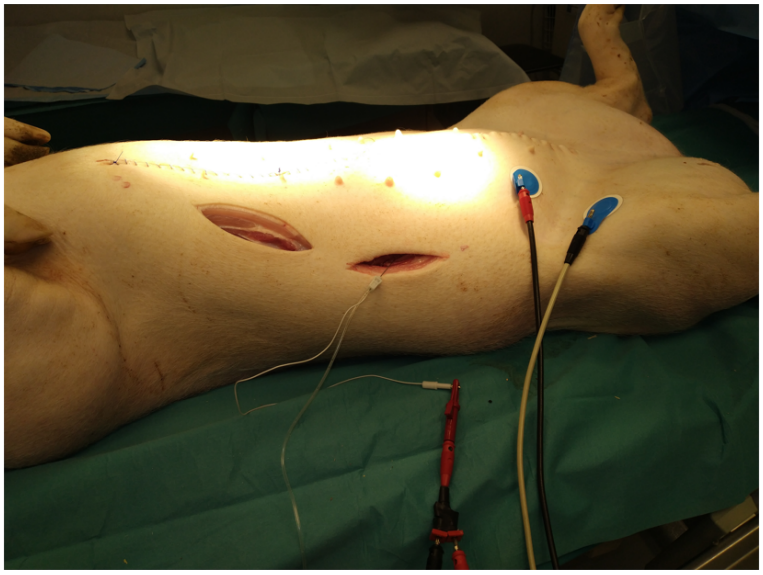

Figure 5.2: The two ECG electrodes attached to the skin of the pig, and the Stimuplex A needle positioned in the tissue exposed by one of the incisions.

As already stated, the parameters specified in LabOne were unchanged from the tests with the Zurich instruments described in chapter 3. The parameters are repeated here

- Frequency range: From 10 Hz to 1 MHz.
- Peak voltage value: 50 mV.
- 50 measurement points on a logarithmic scale, which means 10 measurement points in each decade.

The two newly deceased pigs used during the impedance measurements were quite slim, with a weight of just 56.5 kg and 58,5 kg respectively. Finding enough fat tissue was therefore somewhat challenging. The tissue in the two pigs was also less firm than the tissue in the pork chops and the pieces of side meat used in the test sessions in chapter 3. The lack of firmness made it difficult for an untrained person to find stable positions for the needle. My co-supervisor has long experience with impedance measurements in vivo, so he eventually took over the work of finding positions for the needle in order to save time.

5.3 The two data sets

Five sets, each with 20 impedance spectra, were established in each of the two tissue types with the Stimuplex A needle and the SonoPlex needle. In total that became 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue, which gave a data set with 200 impedance spectra for each of the two needle types prior to the analyzes with the ANNs described in chapter 4. Figure (5.3) shows the impedance spectra established with the Stimuplex A needle and figure (5.4) shows the spectra established with the SonoPlex needle. It is possible to see a pattern in the higher part of the frequency range which distinguish between fat tissue and muscle tissue in the spectra established with the Stimuplex A needle. A corresponding pattern which makes it possible to distinguish between fat tissue and muscle tissue visually, isn't clear in the spectra established with the SonoPlex needle. In general, the spectra from each of the two tissue types look quite different for the two needle types throughout the whole frequency range. The difference suggest that Electrode Polarization Impedance (EPI) is present throughout the whole frequency range, and that the EPI is different for the two needle types. The influence of EPI on the impedance measurements is discussed in chapter 1.

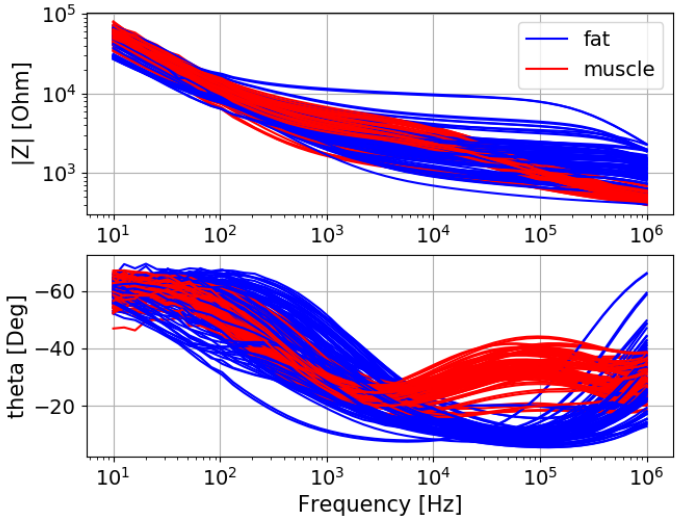

Figure 5.3: 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue established with the Stimuplex A needle.

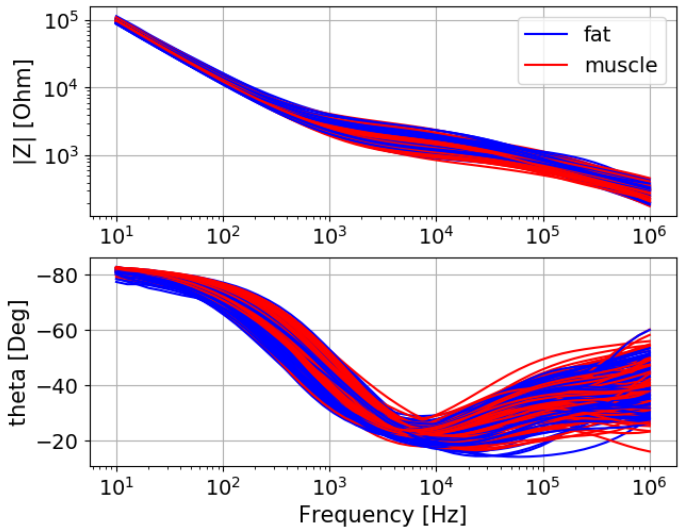

Figure 5.4: 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue established with the SonoPlex needle.

Figure (5.5) and figure (5.6) show four arbitrary impedance spectra from the 100 spectra in fat tissue and the 100 spectra in muscle tissue. Figure (5.6) shows that impedance spectra established with the Stimuplex A needle are disturbed by noise between 10 Hz and 100 Hz. Chapter 3 describes the tests performed in order to find a way to reduce the unwanted noise in the spectra. With a powerbank instead of the power grid as the the power source for the Zurich instrument, the unwanted noise was reduced to a minimum. Therefore it was unexpected to see more noise in in the spectra established with the Stimuplex A needle in figure (5.6) than in the spectra from the final test in side meat tissue shown in figure 3.28. The impedance spectra established with the SonoPlex needle in (5.5) aren't disturbed by noise. The only known difference between two afternoon/evenings with impedance measurements, was the powerful surgical light above the deceased pig. It was turned on during the afternoon/evening when the Stimuplex A needle was used and it was turned off during the afternoon/evening when the SonoPlex needle was used. (The image of the pig taken during the measurement with the Stimuplex A needle in figure (5.2), is disturbed by the surgical light.)

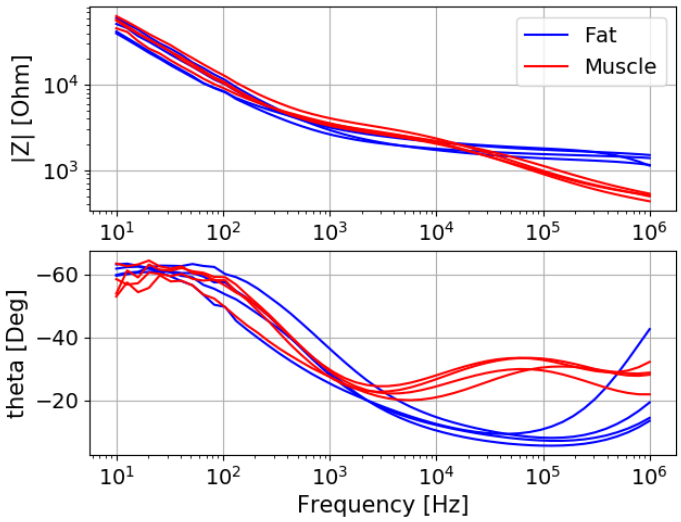

Figure 5.5: Four impedance spectra in fat tissue and four spectra in muscle tissue established with the Stimuplex A needle.

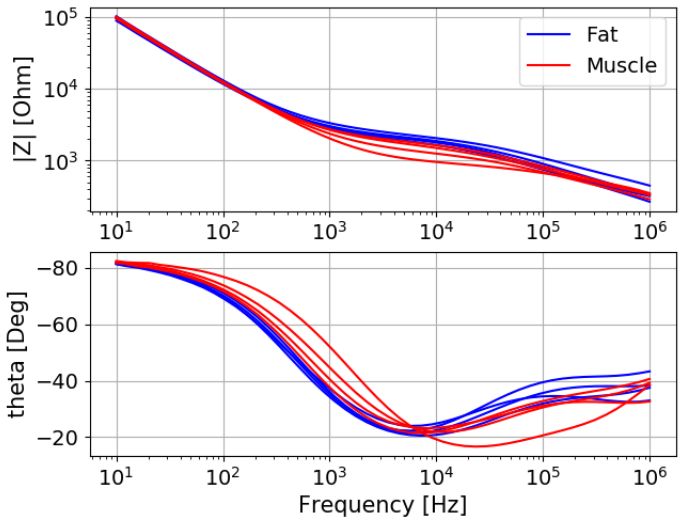

Figure 5.6: Four impedance spectra in fat tissue and four spectra in muscle tissue established with the SonoPlex needle.

Figure (5.7) take a closer look at the frequency range between 10 Hz and 104.8 Hz, where the four impedance impedance spectra in fat tissue and the four spectra in muscle tissue in (5.5) are disturbed by noise. The spectra are scaled by dividing the modulus $|Z|$ with 60000 Ω and the phase angle with 65 degrees. The scale for $|Z|$ in the figure is linear, not logarithmic. Figure (5.8) shows the the real component Z_x and the imaginary component Z_y of these eight impedance spectra in the frequency range 10 Hz and 104.8 Hz. The spectra are scaled by dividing Z_x with 30000 Ω and Z_y with 50000 Ω. The scale for Z_x and Z_y in the figure is linear, not logarithmic. Now the noise in figure (5.7) and figure (5.8) can be compared. The two figures suggest that the calculation of the phase angle amplifies the unwanted noise in Z_x and Z_y, which basically is quite low and considered to be acceptable with regards to the analyzes with the ANNs. Therefore the best option is to represent the impedance spectra with Z_x and Z_y in the analyzes with the ANNs, at least for the spectra established with the Stimuplex A needle. Another option would be to exclude the first frequency decade with $|Z|$ and θ from the analyzes.

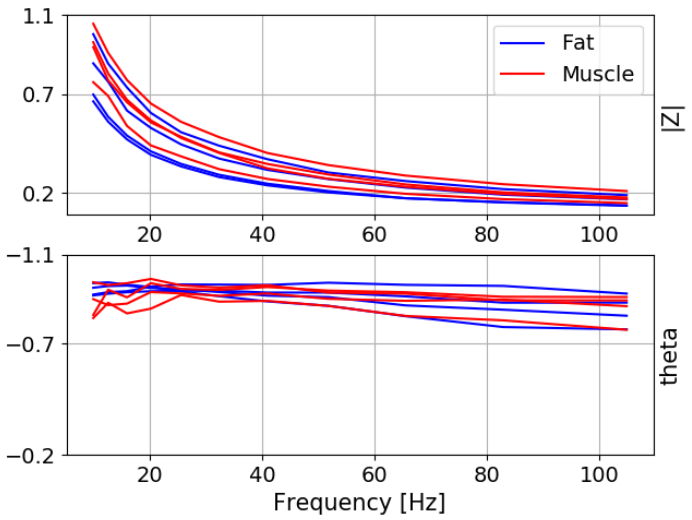

Figure 5.7: The eight impedance spectra established with the Stimuplex A needle in fat and muscle tissue between 10 Hz and 104.8 Hz. The modulus $|Z|$ and the phase angle θ are scaled.

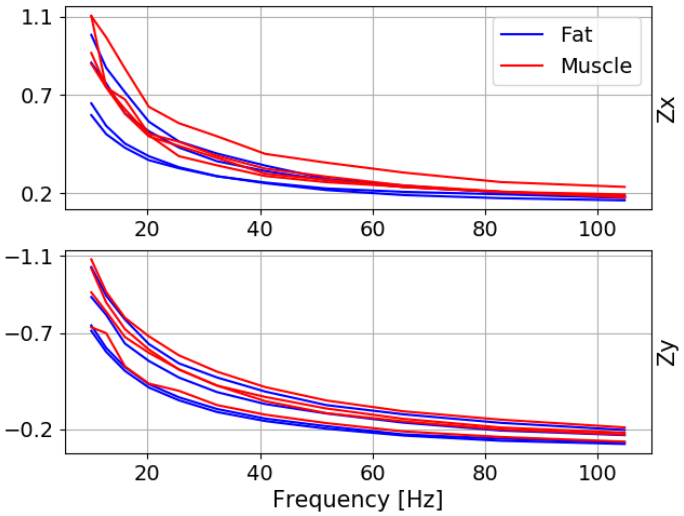

Figure 5.8: The real and the imaginary component of the eight impedance spectra established with the Stimuplex A needle in fat and muscle tissue between 10 Hz and 104.8 Hz. The two components are scaled.

If the noise disturbance only was present in one of the two components, either only in Z_x or only in Z_y, the component disturbed by noise could be replaced by using a discrete version of the analytic expressions of the Kramers-Kronig Transforms given in [6]. The book states that there is a relationship between the real and the imaginary component which makes it possible to calculate the unknown component, which can be either Z_x or Z_y, from the known component. In theory the frequency range should be from 0 Hz and towards infinity. In practice, when the frequency range is limited, some error will occur in the calculations of the unknown component. Since the noise isn't limited to only one of the two components Z_x and Z_y in the impedance spectra, Kramers-Kronig Transforms can't be used to eliminate the noise. Anyways, finding a good discrete approximation of the analytic expressions, and also finding a good estimation of the error in the transformation, would not be straight forward.

Figure (5.9) shows the real component Z_x and the imaginary component Z_y of the 100 impedance spectra in fat tissue and 100 impedance spectra in muscle tissue established with the Stimuplex A needle. Figure (5.10) shows the 200 spectra established with the SonoPlex needle. Since a logarithmic

scale are used for the impedance spectra, the absolute value of the imaginary component is shown. As in figure (5.3), it is possible to see a pattern in the higher part of the frequency range which distinguish between fat tissue and muscle tissue in the spectra established with the Stimuplex A needle. As in figure (5.4), a corresponding pattern which makes it possible to distinguish between fat tissue and muscle tissue visually, isn't clear in the spectra established with the SonoPlex needle.

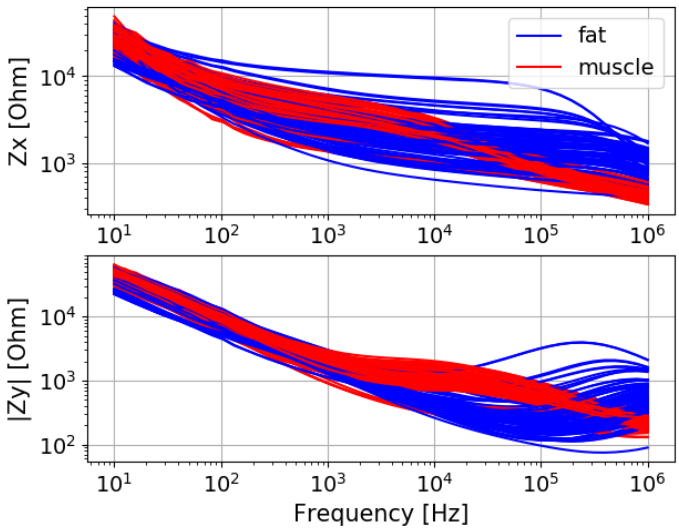

Figure 5.9: The 100 impedance spectra in fat tissue and the 100 impedance spectra in muscle tissue established with the Stimuplex A needle.

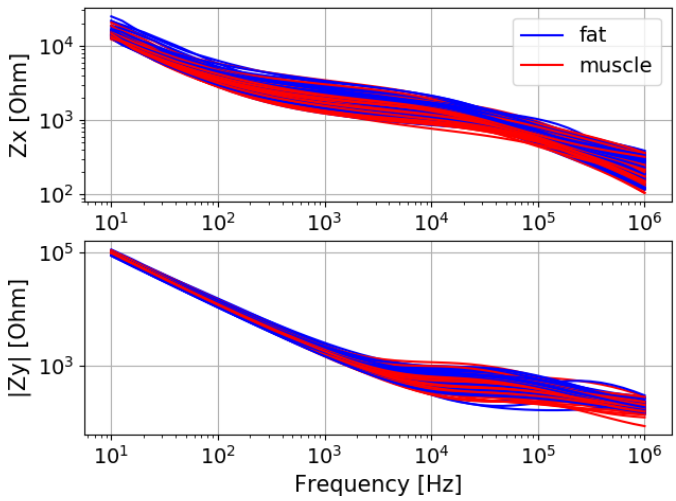

Figure 5.10: The 100 impedance spectra in fat tissue and the 100 impedance spectra in muscle tissue established with the SonoPlex needle.

Chapter 4 covers the organization of the data sets. Each impedance spectrum is called a data example. The data examples in each of the two data sets are organized as a matrix $\mathbf{X}_{data}$, where each row $\mathbf{x}$ is a data example. Each column in $\mathbf{X}_{data}$ is called a feature. The training process of the ANNs is supervised, which means that it is known during the training process which tissue type each of the data examples represent. The known tissue type for a data example is represented by an integer y called the target. Muscle tissue is represented by $y = 0$ and fat tissue is represented by $y = 1$. A data example $\mathbf{x}$ and the corresponding target y of that example is called a pair. The targets are collected in a vector $\mathbf{y}_{data}$. In order to help the ANNs with more information in the training process, the discrete values of the derivatives of real component Z_x and the imaginary component Z_y were included. Each data example then contain 50 Z_x values, followed by 49 values of the discrete derivative of Z_x, 50 Z_y values and 49 values of the discrete derivative of Z_y. In a similar way, the discrete derivatives of the modulus $|Z|$ and the phase angle θ could be included.

In a Feed-forward Neural Network (FNN), the impedance measurement values in each impedance spectrum enter the network simultaneously. In a Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM)

layers/cells, the measurement values in each spectrum enter the RNN in steps which are called time steps or recurrent steps. The measurement values in each spectrum are reorganized as a 50 x 4 matrix. The first column consists of the 50 Z_x values, followed by its 49 values of the discrete derivative in the second column. In order for the second column to contain 50 values, discrete derivative value number 49 is added an extra time as value number 50. The third column consists of the 50 Z_y values, followed by its 49 values of the discrete derivative in the fourth column. Discrete derivative value number 49 is added an extra time as value number 50. Each spectrum enters the RNN in one row at the time from four input nodes in 50 time steps. Each spectrum can also enter the RNN in 198 time steps from one input node as a 198 x 1 vector, but 198 time steps instead of 50 time steps slows down the training process significantly. Since there is no physical connection between the last value in a column and the first value in the next column, using four input nodes seem to be the best choice.

In chapter 1 the influence of Electrode Polarization Impedance (EPI) on the impedance spectra is discussed. The impedance measured with the needle electrode contains EPI in series with the tissue impedance, and the influence from the EPI on the measured impedance is frequency dependent. [5] refers to several articles concerning EPI influence on impedance measurements, and from these sources the conclusion drawn is that considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz. By also dividing each data set into a set containing the impedance measurements at the lower part of the frequency range and a set containing the measurements at the higher part of the frequency range, and analyzing them separately, it is possible to assess if the EPI contains information which can be used in order to distinguish between the fat tissue and the muscle tissue. Each of these two data sets then contains 25 Z_x values, followed by 24 values of the discrete derivative of Z_x, 25 Z_y values and 24 values of the discrete derivative of Z_y. Since the number of features are the same in both data sets, the results for the to data sets from the ANNs can be compared. The 25 impedance measurements points at the lower part of the frequency range are between 10 Hz and 2812 Hz, distributed on a logarithmic scale. The 25 impedance measurements points at the higher part of the frequency range are between 3556 Hz and 1 MHz, distributed on a logarithmic scale. For the RNN with the LSTM layers/cells, the data examples of these two data sets are reorganized as 25 x 4 matrices. The first column of these matrices consists of the 25 Z_x values, followed by its 24 values of the discrete derivative in the second column. The third column

consists of the 25 Z_y values, followed by its 24 values of the discrete derivative in the fourth column. In the to columns with discrete derivative values, value number 24 is added an extra time as value number 25.

Before the training of an ANN starts, each data set is divided into a training set with training pairs and a test set with test pairs. The test set is needed in order to control that the training has worked. The training pairs and the test pairs are randomly chosen from the pairs in the data set.

Chapter 6

Results

6.1 Introduction

In this chapter, the results from the training and testing of the Artificial Neural Networks (ANNs) are presented. The ANNs were explained in chapter 4, and they are

- A Feed-forward Neural Network (FNN) programmed in Python without the support of the Tensorflow 2 and Keras software packages.
- A Feed-forward Neural Network (FNN) implemented by using Tensorflow 2 with the Keras API.
- A Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells, implemented by using Tensorflow 2 with the Keras API. In Tensorflow 2 and Keras a LSTM cell is called a LSTM layer.

For all the three networks, some functionality from the Python machine learning library Scikit-Learn are included in order to prepare the two data sets for the training and testing. The purpose with the FNN programmed in Python was to get an understanding of the terminology used and an idea about what is going on behind the scene in an ANN. This basic FNN performed well in the test with the MNIST data set, which is a part of the Scikit-Learn library and which contains 1797 digital images of hand written numbers between 0 and 9, see 4. Compared to the FNN implemented by using Tensorflow 2 and Keras, its internal structure is quite simple. It is also quite slow, so testing combinations of hyper-parameters in order to optimize the results, will take a long time compared to the FNN implemented with Tensorflow 2 and Keras. Therefore this chapter omits this basic FNN.

The chapter covers the results from the training and testing of the FNN implemented in Tensorflow 2 and Keras, and the RNN with LSTM layers (cells).

In chapter 5, the two data sets which were used in the training and the testing of the ANNs are presented. Each of the two data sets contains 100 impedance spectra from fat tissue and 100 impedance spectra from muscle tissue in a newly deceased pig. One data set was established with a Stimuplex A needle electrode and the other data set was established with a SonoPlex needle electrode. Each data set was established during a separate afternoon/evening at Rikshospitalet in Oslo. A data set with just 200 data pairs is small considering training and testing of neural networks. The MNIST data set which is a part of the Scikit-Learn library, and which is used for training and testing of the ANNs in chapter 4, contains 1797 digital images of hand written numbers between 0 and 9. Each image contains 8 x 8 = 64 pixels. This is also a data set with a modest size considering training and testing of ANNs. (According to [18], the MNIST data set which is included in Tensorflow 2, contains a training set with 60000 digital images of hand written numbers and a test set of 10000 images. Each image contains 28 x 28 pixels). With a data set with only 200 data pairs, there is a risk that the ANNs can be a specialist on classifying the training examples in the training set. They might become over-fitted to the training data, which leads to poorer performance during testing. The tools against over-fitting, which are covered in chapter 4, are L2 regularization and dropout regularization.

Three analyzes were performed with the ANNs for both data sets

- One analysis where the data set contains the features in the 50 measurement points in the frequency range between 10 Hz and 1 MHz.
- One analysis where the data set contains the features in the 25 measurement points in the lower part of the frequency range between 10 Hz and 2812 Hz.
- One analysis where the data set contains the features in the 25 measurement points in the higher part of the frequency range between 3556 Hz and 1 MHz.

The organization of the two data sets for these three analyzes are described in more detail in chapter 5. For each analysis the data set is divided into a training set with 160 training pairs and a test set with 40 test pairs. The training pairs and the test pairs are randomly chosen from the pairs in the

data set. Since the two data sets are small, the random selection of the training pairs and test pairs are seeded in order to be able to compare the results properly The chosen seed is random_state = 1.

6.2 The FNN

An outline of a FNN using Tensorflow 2 and Keras is given in section 4.3.5 in chapter 4. In the outline, only one hidden Dense layer was used. Here two hidden Dense layers are used. The Dense layer, together with some of its arguments, is described in section 4.3.1. With similar combinations of hyper-parameters, initial tests with the SonoPlex data set showed that two hidden layers instead of one gave improvements in the results. The number of nodes chosen are the same in both hidden layers. The activation function is the ReLU activation function, where the parameter a can be chosen. The method used for initialization of weights and bias weights is Glorot Normal. Glorot Uniform is the default method in Tensorflow 2 and Keras for the initialization of the weights in Dense layers, but initial tests with the SonoPlex data set showed slightly better results with Glorot Normal. A separate Dropout layer prior to the two hidden layers and to the output layer initially had dropout rate = 0. After the combination of the other hyper-parameters was chosen, some additional combinations of dropout rates were tested. Combinations of dropout rates were tested separately in order to see if the test result with the chosen combination of the other hyper-parameters could be improved if necessary. Testing with dropout rates were mainly relevant for the data set established with the SonoPlex needle. During the training and testing of the FNN, the result from the training with the training set was very good, but for the data set established with the SonoPlex needle, the testing with the test set indicated over-fitting. Problems with over-fitting should perhaps not be a surprise for a data set with only 200 data pairs. The tools against over-fitting, which are covered in chapter 4, are L2 regularization and dropout regularization.

The 216 combinations of the following hyper-parameters were tested. The number of epochs during the tests was 60.

- Number of nodes in both hidden layers: 50, 100 and 150
- Batch size: 5, 10 and 20
- The ReLU activation function parameter a: 0 and 0.1

- The learning rate (or step size) α with default value $\alpha = 0.001$: 0.0001, 0.001, 0.01
- The regularization parameter λ with default value $\lambda = 0.01$: 0.0001, 0.001, 0.01 and 0.1

In order to get an overview over the most promising combinations, an average of the cost (loss) function, the training accuracy and the test accuracy for the last 10 epochs were calculated for all the combinations. In the figures, which shows the training and testing result for the parameter combination assessed to be the best, the number of epochs are increased to 120 in order to see the further development of the training and testing accuracy beyond 60 epochs. Figures with the cost function are not included.

6.2.1 The Stimuplex A data set

Figure (6.1) to figure (6.3) show the training and test accuracy in percent for the three FNN classification analyzes of the impedance spectra in the data set established with the Stimuplex A needle electrode. For the three analyzes the following combination, out of the 216 combinations tested, was chosen: The number of nodes in both hidden layers were 50, the batch size was 10, ReLU parameter a = 0.1, $\alpha = 0.0001$ and $\lambda = 0.01$. Several other combinations gave equally good results. A dropout rate larger than zero in one, two or three of the dropout layers wasn't included.

Here is a description of the results in the three figures

- Figure (6.1) show the results for the data set when the impedance spectra extend over the whole frequency range between 10 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 97.5 percent.
- Figure (6.2) show the results for the data set when the impedance spectra are limited to the frequency range between 10 Hz and 2812 Hz. The average training accuracy for the last 10 epochs is 99.7 percent and the average test accuracy for the last 10 epochs is 95.0 percent.
- Figure (6.3) show the results for the data set when the impedance spectra are limited to the frequency range between 3556 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 100 percent.

So the results are very good. Electrode Polarization Impedance (EPI) doesn't seem to have any negative impact on the results from the lower part of the frequency range compared to the results from the higher part of the frequency range. One small difference worth noticing perhaps, is that more epochs are needed for the training results to reach close to 100 percent accuracy in the analysis where the frequency range is between 10 Hz and 2812 Hz.

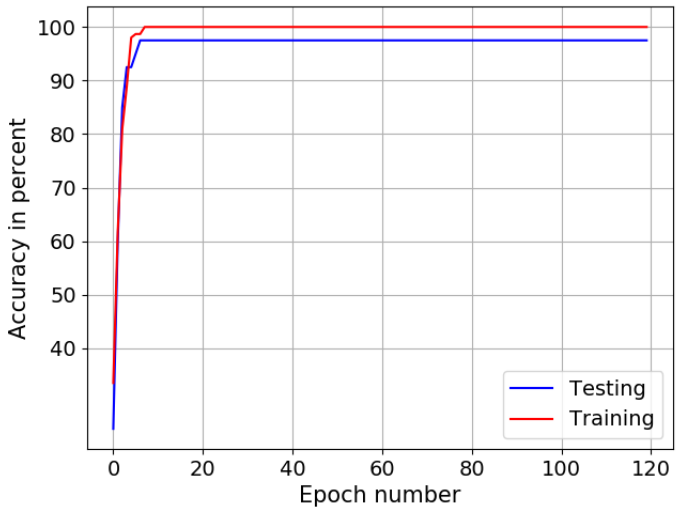

Figure 6.1: The training and testing accuracy when the impedance spectra extend between 10 Hz and 1 MHz. Needle: Stimuplex A.

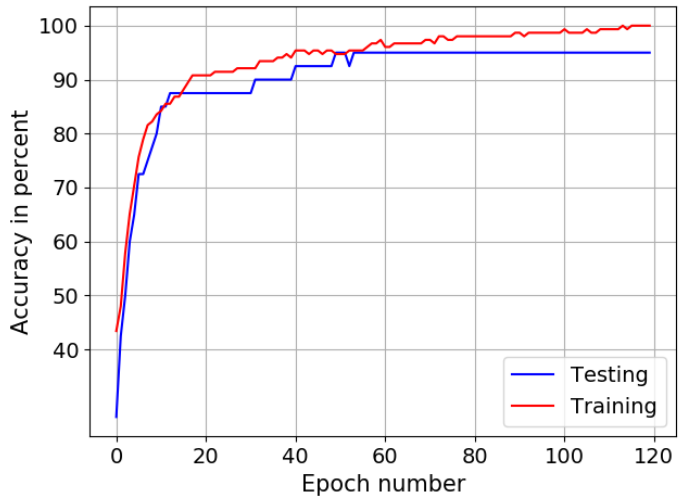

Figure 6.2: The training and testing accuracy when the impedance spectra extend between 10 Hz and 2812 Hz. Needle: Stimuplex A.

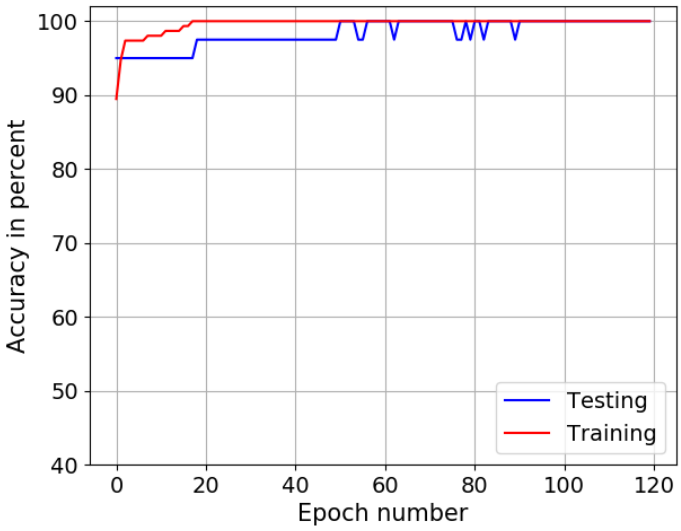

Figure 6.3: The training and testing accuracy when the impedance spectra extend between 3556 Hz and 10 MHz. Needle: Stimuplex A.

6.2.2 The SonoPlex data set

Figure (6.4) to figure (6.6) show the training and test accuracy in percent for the FNN classification analyzes of the impedance spectra in the data set established with the SonoPlex needle electrode. For the three analyzes the following three combinations, out of the 216 combinations tested, were assessed to be the best

- For the analysis with frequency range between 10 Hz and 1 MHz, the number of nodes in both hidden layers were 150, the batch size was 5, ReLU parameter a = 0.1, $\alpha = 0.0001$ and $\lambda = 0.001$.
- For the analysis with frequency range between 10 Hz and 2812 Hz, the number of nodes in both hidden layers were 100, the batch size was 10, ReLU parameter a = 0.1, $\alpha = 0.001$ and $\lambda = 0.0001$.
- For the analysis with frequency range between 3556 Hz and 1 MHz, the number of nodes in both hidden layers were 100, the batch size was 20, ReLU parameter a = 0.1, $\alpha = 0.001$ and $\lambda = 0.001$.

Here is a description of the results in the three figures

- Figure (6.4) show the results for the data set when the impedance spectra extend over the whole frequency range between 10 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 87.5 percent.
- Figure (6.5) show the results for the data set when the impedance spectra are limited to the frequency range between 10 Hz and 2812 Hz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 80.0 percent.
- Figure (6.6) show the results for the data set when the impedance spectra are limited to the frequency range between 3556 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 98.4 percent and the average test accuracy for the last 10 epochs is 79.0 percent.

The training results are good for all the three frequency ranges. In the analysis with frequency range between 3556 Hz and 10 MHz, the progress in the training is a bit unstable, and the training result is not quite as good

as in the analysis with frequency range between 10 Hz and 2812 MHz. The test results are not as good as the training results. Including a dropout rate larger than zero in one, two or three of the dropout layers didn't improve the testing results. Instead the improvement in training accuracy as the training progressed through the epochs became slower and the training results had a tendency to become less accurate at the end of the training. The test results are better when the impedance spectra extend over the whole frequency range, than when the spectra are limited to the frequency range between 10 Hz and 2812 Hz or limited to the range between 3556 Hz and 1 MHz. The influence of EPI doesn't have any effect on the results from the lower part of the frequency range compared to the results from the higher part of the frequency range.

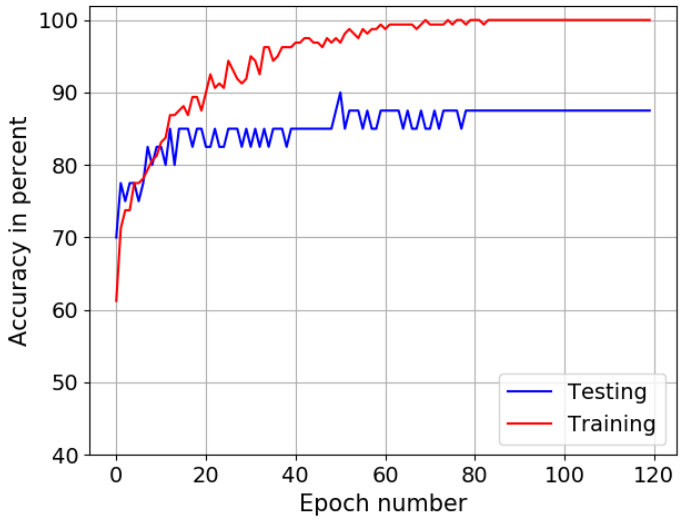

Figure 6.4: The training and testing accuracy when the impedance spectra extend between 10 Hz and 1 MHz. Needle: SonoPlex.

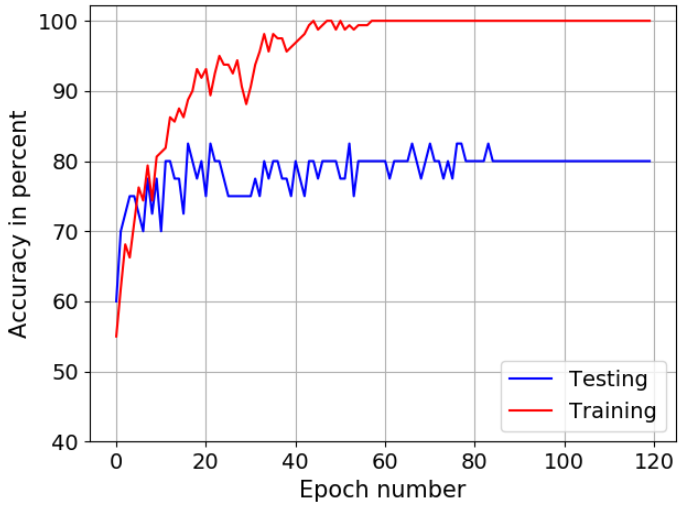

Figure 6.5: The training and testing accuracy when the impedance spectra extend between 10 Hz and 2812 Hz. Needle: SonoPlex.

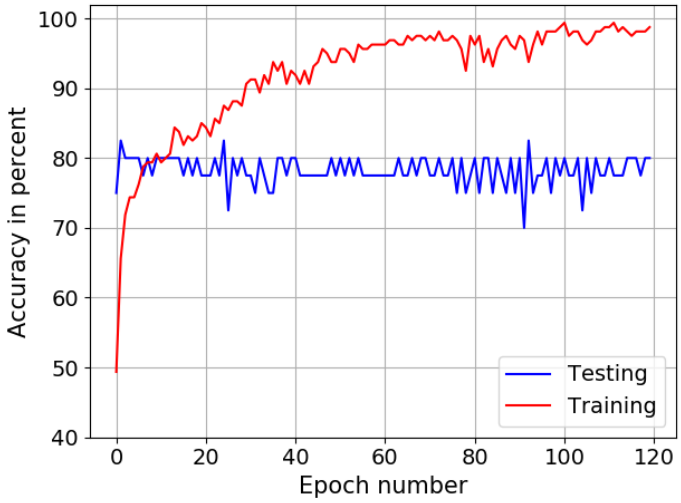

Figure 6.6: The training and testing accuracy when the impedance spectra extend between 3556 Hz and 10 MHz. Needle: SonoPlex.

6.3 The RNN with LSTM layers (cells)

An outline of a RNN with LSTM layers (cells) using Tensorflow 2 and Keras is given in section 4.4.1 in chapter 4. The RNN with LSTM layers are explained in section 4.4. In the outline, only one LSTM layer was used. Here two LSTM layers are used. With similar combinations of hyper-parameters, initial tests with the SonoPlex data set showed that two LSTM layers instead of one gave improvements in the results. The number of nodes chosen were the same in both LSTM layers. The default activation functions were used, which is the hyperbolic tangent function as the activation function and the sigmoid function as the recurrent activation function. The method chosen for the initialization of the weights was the default initializer method, which is Glorot Uniform. The method chosen for the initialization of the recurrent weights was the default recurrent initializer method, which is Orthogonal. Initialization of weights, together with some of the methods, is described in conjunction with the Dense layer in section 4.3.1. In initial tests with the SonoPlex data set, two other method combinations were also tested. One was with Glorot Normal as both the initializer method for the weights and for the recurrent weights. The other was with Glorot Normal as the initializer method for the weights and Orthogonal as the initializer method for the recurrent weights. The tests showed slightly better results with the two default initializer methods, Glorot Uniform and Orthogonal. Instead of using separate Dropout layers, an argument in each of the two LSTM layers for the dropout rate and the recurrent dropout rate respectively can be used. Initially the dropout rates were set to zero. After the combination of the other hyper-parameters was chosen, some additional combinations of dropout rates were tested. Combinations of dropout rates were tested separately in order to see if the test result with the chosen combination of the other hyper-parameters could be improved if necessary. Testing with dropout rates were mainly relevant for the data set established with the SonoPlex needle. During the training and testing of the RNN, the result from the training with the training set was very good. For the data set established with the SonoPlex needle, the testing with the test set indicated over-fitting in the same way as in the testing of the FNN with this data set.

Compared to the FNN, 81 combinations of hyper-parameters were tested instead of 216. The number of combinations is reduced from 216 to 108 because the ReLU activation function isn't used. $\lambda = 0.1$ is omitted because this value of the parameter gave poor results compared to the other choices of this parameter in the tests with the FNN. The number of combinations

is then reduced from 108 to 81.

- Number of nodes in both hidden layers: 50, 100 and 150
- Batch size: 5, 10 and 20
- The learning rate (or step size) α with default value $\alpha = 0.001$: 0.0001, 0.001, 0.01
- The regularization parameter λ with default value $\lambda = 0.01$: 0.0001, 0.001 and 0.01

Due to the time steps, or recurrent steps, it takes considerable longer time to run through one single combination of hyper-parameters in the RNN with LSTM layers than in the FNN. In order to get an overview over the most promising combinations, an average of the cost (loss) function, the training accuracy and the test accuracy for the last 10 epochs were calculated for all the combinations. In the figures, which shows the training and testing result for the parameter combination assessed to be the best, the number of epochs are increased to 120 in order to see the further development of the training and testing accuracy beyond 60 epochs.

6.3.1 The Stimuplex A data set

Figure (6.7) to figure (6.9) show the training and test accuracy in percent for the three RNN classification analyzes of the impedance spectra in the data set established with the Stimuplex A needle electrode. For the three analyzes the following combination, out of the 81 combinations tested, was chosen: The number of nodes in both hidden layers were 50, the batch size was 10, $\alpha = 0.001$ and $\lambda = 0.0001$. Several other combinations gave equally good results. A dropout rate larger than zero in one, two or three of the dropout layers wasn't included.

Here is a description of the results in the three figures

- Figure (6.7) show the results for the data set when the impedance spectra extend over the whole frequency range between 10 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 97.5 percent.
- Figure (6.8) show the results for the data set when the impedance spectra are limited to the frequency range between 10 Hz and 2812

Hz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 95.0 percent.

- Figure (6.9) show the results for the data set when the impedance spectra are limited to the frequency range between 3556 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 100 percent.

So the results are very good. Electrode Polarization Impedance (EPI) doesn't seem to have any negative impact on the results from the lower part of the frequency range compared to the results from the higher part of the frequency range. One small difference worth noticing perhaps, is that the test accuracy is a little lower in the analysis where the frequency range is between 10 Hz and 2812 Hz.

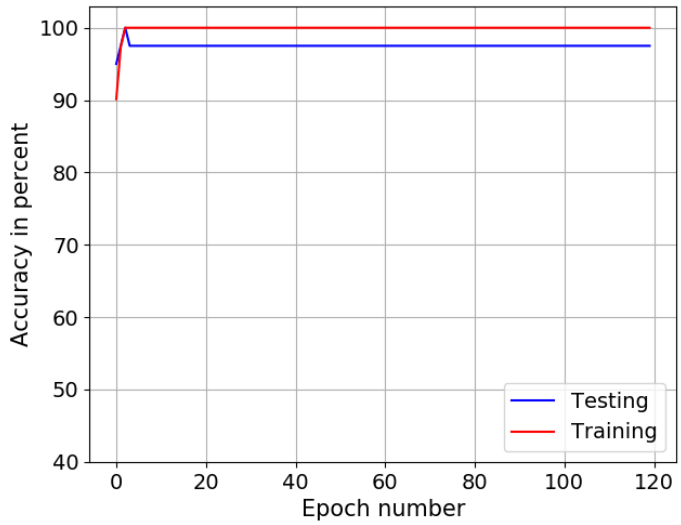

Figure 6.7: The training and testing accuracy when the impedance spectra extend between 10 Hz and 1 MHz. Needle: Stimuplex A.

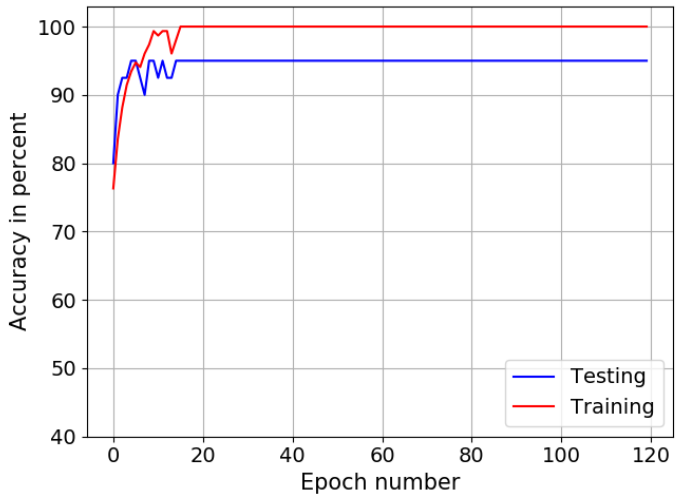

Figure 6.8: The training and testing accuracy when the impedance spectra extend between 10 Hz and 2812 Hz. Needle: Stimuplex A.

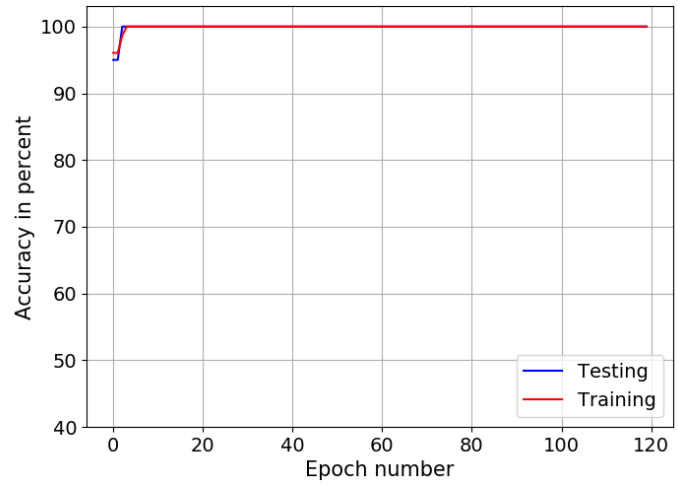

Figure 6.9: The training and testing accuracy when the impedance spectra extend between 3556 Hz and 10 MHz. Needle: Stimuplex A.

6.3.2 The SonoPlex data set

Figure (6.10) to figure (6.12) show the training and test accuracy in percent for the RNN classification analyzes of the impedance spectra in the data set established with the SonoPlex needle electrode. For the three analyzes the following three combinations, out of the 81 combinations tested, were assessed to be the best

- For the analysis with frequency range between 10 Hz and 1 MHz, the number of nodes in both hidden layers were 100, the batch size was 10, $\alpha = 0.001$ and $\lambda = 0.0001$.
- For the analysis with frequency range between 10 Hz and 2812 Hz, the number of nodes in both hidden layers were 100, the batch size was 5, $\alpha = 0.001$ and $\lambda = 0.0001$.
- For the analysis with frequency range between 3556 Hz and 1 MHz, the number of nodes in both hidden layers were 50, the batch size was 20, $\alpha = 0.01$ and $\lambda = 0.001$.

Here is a description of the results in the three figures

- Figure (6.10) show the results for the data set when the impedance spectra extend over the whole frequency range between 10 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 82.5 percent.
- Figure (6.11) show the results for the data set when the impedance spectra are limited to the frequency range between 10 Hz and 2812 Hz. The average training accuracy for the last 10 epochs is 100 percent and the average test accuracy for the last 10 epochs is 77.0 percent.
- Figure (6.12) show the results for the data set when the impedance spectra are limited to the frequency range between 3556 Hz and 1 MHz. The average training accuracy for the last 10 epochs is 98.5 percent and the average test accuracy for the last 10 epochs is 77.0 percent.

The training results are good for all the three frequency ranges. Compared with the results from the analyzes with the FNN in figure (6.4) to figure (6.6), both the training and the testing accuracy are more unstable as the

training and testing progress through the epochs. The test results are not as good as the training results. The RNN doesn't reduce the difference between training accuracy and testing accuracy compared to the results from the analyzes with the FNN. A few experiments with combinations of the dropout rate and the recurrent dropout rate with values larger than zero in the two LSTM layers, didn't improve the testing results. Instead the results from the training became less accurate. The results are slightly better when the impedance spectra extend over the whole frequency range, than when the spectra are limited to the frequency range between 10 Hz and 2812 Hz or limited to the range between 3556 Hz and 1 MHz. The influence of EPI doesn't have any effect on the results from the lower part of the frequency range compared to the results from the higher part of the frequency range.

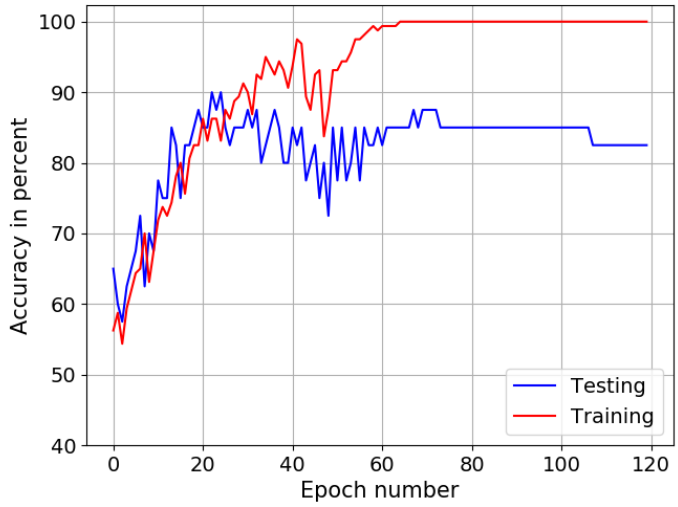

Figure 6.10: The training and testing accuracy when the impedance spectra extend between 10 Hz and 1 MHz. Needle: SonoPlex.

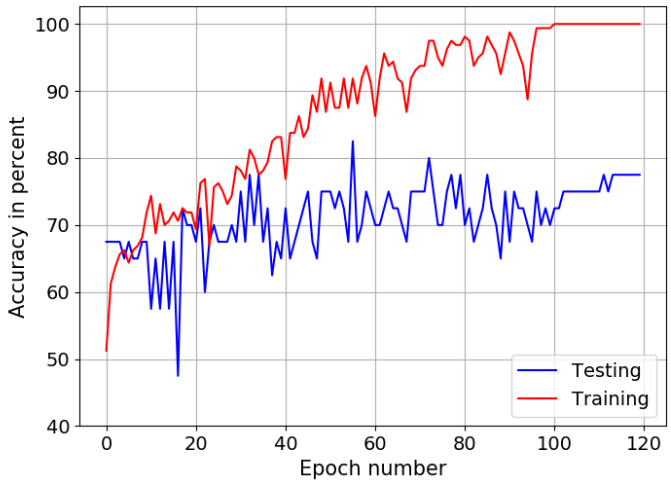

Figure 6.11: The training and testing accuracy when the impedance spectra extend between 10 Hz and 2812 Hz. Needle: SonoPlex.

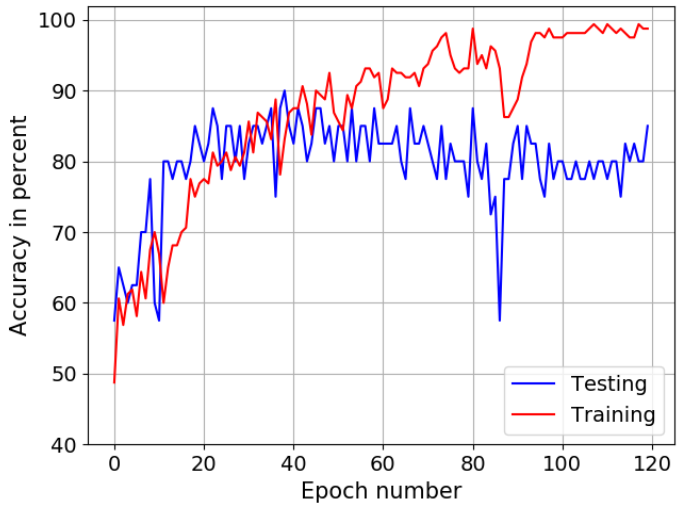

Figure 6.12: The training and testing accuracy when the impedance spectra extend between 3556 Hz and 10 MHz. Needle: SonoPlex.

Chapter 7

Summary and conclusion

7.1 Summary

The foundation of this thesis is the work performed by Kalvøy et al in [5]. In this article, two needle electrode types were used to establish impedance spectra through impedance measurements in different tissue types in a pig under anesthesia. The frequency range of the spectra was between 10 Hz and 1 MHz. The purpose was to investigate whether the impedance spectra had tissue dependent characteristics which made it possible to distinguish between the spectra established in different tissue types. Plots of the impedance modulus $|Z|$ and phase angle θ of the spectra in fat tissue and muscle tissue showed different patterns for the two tissue types at frequencies above 100 kHz. A partial least-squares discriminate analysis and a regression analysis with the Unscrambler 9.7 software from Camo Inc., showed that the impedance spectra in muscle tissue can be separated from the impedance spectra in fat and subdermal fat tissue. The analyzes were performed on up to 12 impedance spectra in muscle tissue, fat tissue and subdermal fat tissue respectively.

The electrode area of a needle electrode at the tip of the needle is small. As an example, [12] gives a rough estimate of the electrode area of the Stimuplex A needle, see section 3.2 in chapter 3. The estimated electrode area is $0.7\,mm^2$. The Stimuplex A needle was the needle type used to establish one of the two data sets with impedance spectra from fat and muscle tissue in a newly deceased pig in this thesis, see chapter 5. A small electrode area gives a small sensitivity volume. This can be explained with an analytical example with an electrode which has the shape of a hemisphere

or a sphere, see [6] and section 2.4 in chapter 2 in this thesis. It is also shown by an experiment with a needle in a tank with saline in [5] and by a Finite Element Analysis of the needle and the saline in [7]. The experiment, the Finite Element Model and the Finite Element Analysis are briefly described in chapter 1. The sensitivity volume, which is spherical and has its center at the tip of the needle electrode, is introduced in chapter 1. In order to determine the tissue type in the immediate vicinity of the needle tip with the information from the impedance measurements, the radius of the sensitivity volume needs to be small. This is because a small sensitivity volume gives a high probability for the tissue within the volume to be homogeneous.

The small electrode area of the needle electrode entails influence from Electrode Polarization impedance (EPI) on the measured impedance. EPI is explained in chapter 1. For a spherical electrode for example, the ratio between the EPI and the tissue impedance is inversely proportional to the electrode radius, according to [6]. The impedance measured with the needle electrode therefore contains EPI in series with the tissue impedance. The influence from EPI is also frequency dependent, and considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz, see [5] and chapter 1 in this thesis. According to [5], the EPI can be tissue dependent. Instead of just trying to minimize the influence of EPI in the impedance measurements by omitting the frequency range below 50 kHz, the article ([5]) suggests that the EPI in the measurements can be exploited in order to distinguish between different types of tissue.

In this thesis the tissue dependencies of the EPI are explored further. Two data sets with impedance spectra were established, where a needle electrode was positioned in many different locations in fat and muscle tissue in a newly deceased pig. In order to establish one of the sets, a Stimuplex A needle was used. The other set was established with a SonoPlex needle. The impedance measurements for each set were performed on a separate afternoon/evening at Rikshospitalet in Oslo after other researches had completed their research work on the pig when it was under anesthesia, and the pig was put to death. 100 impedance spectra in muscle tissue and 100 impedance spectra in fat tissue in the frequency range between 10 Hz and 1 MHz were established with each of the two needle electrode types. Each impedance spectrum contains 50 measurement points on a logarithmic scale, with 10 points in each of the 5 decades between 10 Hz and 1 MHz. In addition to the real component Z_x and the imaginary component Z_y of the impedance, the discrete derivatives of these two components were calculated and included in the two data sets

in order to help the Artificial Neural Networks (ANNs) during training and testing. The organization of the data in the two data sets are described in section 5.3 in chapter 5. The two data sets were analyzed with two ANNs: One Feed-forward Neural Network (FNN) and one Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells. The two ANNs were implemented by using Tensorflow 2 with the Keras API. In Tensorflow 2 and Keras, a LSTM cell is called a LSTM layer. Three analyzes were performed with the ANNs for both data sets: One analysis in the entire frequency range between 10 Hz and 1 MHz, one analysis in the lower part of the frequency range between 10 Hz and 2812 Hz and one analysis in the higher part of the frequency range between 3556 Hz and 1 MHz. The analyzes in the frequency range between 10 Hz and 2812 Hz are well within the range where considerable influence from EPI must be expected. The lower part and the higher part of the frequency range each contained half of the impedance measurement points, which makes it possible to compare the ANN analyzes from these two ranges and make an assessment of the EPI's influence on the accuracy of the two ANNs.

The chapter prior to this chapter covers the following topics

- Chapter 1, which is the introduction, introduces the sensitivity volume of an electrode, and Electrode Polarization Impedance (EPI).
- Chapter 2 explains electrical impedance, explains the electrode setup which was used during the impedance measurements and covers the analytical example regarding the sensitivity volume of an electrode with the shape of a hemisphere or a sphere.
- Chapter 3 covers the impedance measurements in saline, in two pork chops and in two pieces of side meat, which were performed in preparation for the two evenings/afternoons with impedance measurements in fat and muscle tissue a newly deceased pig at Rikshospitalet in Oslo. Three needle types and two impedance analyzers were tested. The two needle types chosen for the measurements at Rikshospitalet were the Stimuplex A needle and the SonoPlex needle. The instrument chosen was the MFIA impedance analyzer from Zurich Instruments.
- Chapter 4 covers the ANNs. Three ANNs are presented. In order to to get an understanding of the terminology used and an idea about what is going on behind the scene in an ANN, a FNN programmed in Python is covered. Then the FNN and the RNN with LSTM cells (or

layers), which were implemented by using Tensorflow 2 with the Keras API, are covered.

- Chapter 5 covers the impedance measurements in fat and muscle tissue in a newly deceased pig during the two afternoons/evenings at Rikshopitalet in Oslo. Considering the classification analyzes with the ANNs, the chapter also describes the organisation of the two data sets with the established impedance spectra, one set for each of the two needle types used to measure the impedance.

- Chapter 6 covers the results of the training and testing with the two ANNs which were implemented by using Tensorflow 2 with the Keras API.

7.2 Conclusion

Each of the two data sets contain 100 impedance spectra from fat tissue and 100 impedance spectra from muscle tissue. For each analysis the data set is divided into a training set with 160 training pairs and a test set with 40 test pairs. The test set is needed in order to control that the training of the neural network has worked. Figure (5.9) and figure (5.10) in chapter 5 show the impedance spectra in the two data sets. Figure (5.9) shows that it is possible to see a pattern in the higher part of the frequency range which distinguish between fat tissue and muscle tissue in the spectra established with the Stimuplex A needle. A corresponding pattern which makes it possible to distinguish between fat tissue and muscle tissue visually, is difficult to find in the spectra established with the SonoPlex needle. At the higher part of the frequency range, the patterns in the same type of tissue are not similar for the two needle types. It means that the properties of the needle electrode also has an influence on the measured impedance at the higher part of frequency range in the form of Electrode Polarization Impedance (EPI).

Chapter 6 shows very good results from the training of the two ANNs. For the data set established with the SonoPlex needle, the results from the testing are less accurate than the training results. It suggest over-fitting, which means that the two neural networks haven't been able to find a general pattern in the SonoPlex training set which can be transferred to an equally good accuracy during testing with the test set. Instead the two networks seem to have been specialists in classifying the training examples in the SonoPlex

training set. Dropout regularisation, which is a tool against over-fitting, didn't improve the test results. Instead the training results became less accurate. A possible solution is a data set with a larger number of impedance spectra in each tissue type. Then the two networks will have a larger training set to find general patterns in, with better test accuracy as a possible result. A larger data set with more than 100 impedance spectra in each of the two tissue types, requires more time than one afternoon/evening to establish. The test accuracy for the data set established with the Stimuplex A needle are very good, so the two neural networks seem to be able to find a general pattern in the Stimuplex A training set which can be transferred to as good accuracy, or to almost as good accuracy, during testing.

In a FNN, the impedance measurement values (Z_x, Z_y and their discrete derivatives) in each impedance spectrum enter the network simultaneously. In a RNN with LSTM cells (or layers), the impedance measurement values in each spectrum enter the RNN in steps which are called time steps or recurrent steps. The details about how the recurrernt steps are organized, are described in section 5.3 in chapter 5. Since it was quite possible there was a time dependency between the Z_x components, between the Z_y components, between the discrete derivatives of the Z_x components and between the discrete derivatives of the Z_y components, it was assumed that the RNN was able to take advantage of information in the two data sets which was inaccessible for the FNN. For the data set established with the Stimuplex A needle, an improvent of the results from the FNN couldn't be expected, but the RNN still improved the results slightly. For the data set established with the Sonoplex needle, the results from the FNN were more stable through the epochs, and in general better. Therefore the conclusion is that the RNN doesn't seem to find additional information in the two data sets compared to the information available to the FNN. In addition, an analysis with the FNN takes significantly less time.

By comparing the accuracy of the analyzes in the lower part of the frequency range between 10 Hz and 2812 Hz with the accuracy of the analyzes in the higher part of the frequency range between 3556 Hz and 1 MHz, it is possible to make an assessment of the negative influence from EPI on the analyzes in the frequency range between 10 Hz and 2812 Hz. The results presented in chapter 6 don't show any clear signs of negative influence from EPI. There are to possible reasons which can explain that the results from the analyzes in the frequency range between 10 Hz and 2812 Hz are as good as, or almost as good as, the results from the analyzes in the frequency range between

3556 Hz and 1 MHz. The first reason is that the influence from EPI is considerable between 10 Hz and 2812 Hz, and that the the ANNs have been able to find a pattern in the EPI which is dependent of the tissue type. Then the ANN analyzes have been able to confirm that the EPI contains tissue information which can be used in order to distinguish between fat tissue and muscle tissue. The second reason can be that the statement which says that considerable influence from EPI must be expected in the measurements for frequencies up to 10 kHz – 50 kHz, is too conservative. If the influence from EPI only is considerable up to for example 1 kHz in the impedance spectra, there still is a part of the impedance spectra in the frequency range between 10 Hz and 2812 Hz which aren't heavily influenced by EPI. Then it is possible for this part of the spectra to contain enough tissue information for the ANNs to produce results with accuracy similar to the analyzes of the spectra in the range between 3556 Hz and 1 MHz. In order to rule out the second reason or to make it less likely, additional analyzes of the impedance spectra can be limited to, for example, the frequency range between 10 Hz and 1 kHz and to the frequency range between 10 kHz and 1 MHz respectively. Since these two frequency ranges both contain 10 impedance measurement point, the results from the ANNs can be compared.

7.3 Some thoughts on further work

So a continuation of the work in this thesis which isn't difficult to carry through, is to perform separate analyzes with the two data sets already established for the part of the impedance spectra in the frequency range between 10 Hz and 1 kHz and for the part of the spectra in the frequency range between 10 kHz and 1 MHz. By comparing the results from these two frequency ranges, it will be possible to assess whether the EPI in the part of the spectra which is between 10 Hz and 1 kHz contain enough tissue information for the ANNs to produce results with accuracy equal, or almost equal, to the result for the part of the spectra between 10 kHz and 1 MHz. If the accuracy in the results is equal, or almost equal, for the spectra in the two frequency ranges, the range of the two frequency ranges can be limited even more, and the process with ANN analyzes repeated. A source of error in this repeated process is that the number of impedance measurements in the two frequency ranges eventually becomes too low in order to provide enough information for the ANNs.

To expand the work further, especially the data set established with the

SonoPlex needle needs a larger number of impedance spectra. With more impedance spectra in the training set, problems with over-fitting are easier to avoid, and the test results for this data set might be improved. The number of impedance measurement points in each spectrum can also be increased. Then the ANNs will have enough information from each spectrum, even if the frequency ranges in the ANN analyzes are very limited. A data set with a larger number of impedance spectra, where each spectrum contains a larger number of measurement points, will take longer time to establish. If the fat and muscle tissue in a piece of side meat bought at the supermarket was a good replacement for living fat and muscle tissue, establishing a larger data set wouldn't be difficult. Unfortunately impedance measurements in a piece of side meat can't replace measurements in living tissue. Instead a pig, under anesthesia or newly deceased, is required for each day with measurements, and then establishing a larger data set becomes more challenging.

Bibliography

[1] Ling C, de Craen A, Slagboom P, Gunn D, Stokkel M, Westendorp R, Maier A. (2011). Accuracy of direct segmental multi-frequency bioimpedance analysis in the assessment of total body and segmental body composition in middle-aged adult population. Clinical Nutrition, vol. 30 (5), 610-615. Available at: https://www.clinicalnutritionjournal.com/article/S0261-5614(11)00066-5/fulltext

[2] Esco MR, Snarr RL, Leatherwood MD, Chamberlain NA, Redding ML, Flatt AA, Moon JR, Willifield HN. (2015). Comparison of total segmental body composition using DXA and multifrequency bioimpedance in collegiate female athletes. Journal of Strength and Conditioning Research, vol 24 (4), 918-925. Available at: https://journals.lww.com/nsca-jscr/Fulltext/2015/04000/Comparison_of_Total_and_Segmental_Body_Composition.9.aspx

[3] Cao H, Tungjitkusolmun S, Choy YB, Tsai JZ, Vorperian VR, Webster JG. (2002). Using electrical impedance to predict catheter-endocardial contact during RF cardiac ablation. IEEE Transactions on Biomedical Engineering, vol 49 (3), 247-253.
Available at: https://ieeexplore.ieee.org/document/983459

[4] Chizitz JS, Michaud GF, Stephenson K. (2017). Impedance-guided Radiofrequency Ablation: Using Impedance to Improve Ablation Outcomes. The Journal of Innovations in Cardiac Rythm Management, vol. 8 (10), 2868-2873. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7252711/

[5] Kalvøy H, Frich L, Grimnes S, Martinsen ØG, Hol PK, Stubhaug A. (2009). Impedance-based tissue discrimination for needle guidance. Physiological Measurement, vol. 30 (2), 129-140. Available at: https://iopscience.iop.org/article/10.1088/0967-3334/30/2/002

[6] Grimnes S, Martinsen ØG. (2015). Bioimpedance & Bioelectrisity Basics, 3rd ed. Elsevier Ltd.

[7] Høyum P, Kalvøy H, Martinsen ØG, Grimnes S. (2010). A finite element model of needle electrode spatial sensitivity. Physiological Measurement, vol. 31 (10), 1369-1379. Available at: https://iopscience.iop.org/article/10.1088/0967-3334/31/10/006

[8] Schwan HP. (1992). Linear and non-linear electrode polarization and biological materials. Annals of Biological Engineering, vol. 20 (3), 269 - 288. Available at: https://link.springer.com/article/10.1007/BF02368531

[9] Paynter Rt, Boydell BJT. (2009). Electronics Technology Fundamentals, Electron Flow Version, 3rd ed. Pearson Prentice Hall.

[10] Sauerheber R, Heinz B. (2015). Temperature Effects on Conductivity of Seawater and Physiologic Saline, Mechanism and Significance, Chemical Sciences Journal, vol 6 (4). Available at: https://www.hilarispublisher.com/archive/csj-volume-6-issue-4-year-2015.html

[11] Wikipedia. The Free Encyclopedia. Birmingham gauge [online]. Available at: https://en.wikipedia.org/wiki/Birmingham_gauge (Accessed 8th December 2021)

[12] Kalvøy H, Tronstad C, Nordbotten B, Grimnes S, Martinsen ØG. (2010). Electrical Impedance of Stainless Steel Needle Electrodes. Annals of Biomedical Engineering, vol. 38 (7), 2371-2382. Available at: https://www.researchgate.net/publication/41850436_Electrical_Impedance_of_Stainless_Steel_Needle_Electrodes

[13] Tronstad C and Strand-Amundsen R. (2019). Possibilities in the application of machine learning on bioimpedance time-series. Journal of Electrical Bioimpedance, vol. 10 (1), 24-33. Available at: https://doi.org/10.2478/joeb-2019-0004

[14] Strand-Amundsen RJ, Tronstad C, Reims HM, Reinholt FP, Høgetveit JO, Tønnessen TI. (2018). Machine learning for intraoperative prediction of viability in ischemic small intestine. Physiological Measurement,

vol 39 (10): 105011.
Available at: https://doi.org/10.1088/1361-6579/aae0ea

[15] Cunha AB, Hou J, Schuelke C. (2019). Machine learning for stem cell differentiation and proliferation classification on electrical impedance spectroscopy. Journal of Electrical Bioimpedance, vol. 10 (1), 124-132. Available at: https://doi.org/10.2478/joeb-2019-0018

[16] Hastie T, Tibshirani R, Friedman J. (2009). The Elements of Statistical Learning, 2nd ed. New York: Springer. ISBN: 978-0-387-84857-0

[17] Thomas A. (2017). Neural Networks Tutorial - A Pathway to Deep Learning. Adventures in Machine Learning [online].
Available at:
https://adventuresinmachinelearning.com/neural-networks-tutorial/
About Thomas A:
https://adventuresinmachinelearning.com/about/
(Accessed 10th September 2021)

[18] Thomas A. Coding the Deep Learning Revolution. Adventures in Machine Learning [online]. Available at:
https://adventuresinmachinelearning.com/product/
coding-the-deep-learning-revolution/
About Thomas A:
https://adventuresinmachinelearning.com/about/
(Accessed 10th September 2021)

[19] Harrison. Deep Learning basics with Python, Tensorflow and Keras. Pythonprogramming.net [online]. Available at:
https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/
(Accessed 10th September 2021)

[20] Thomas A. (2017). Improve your neural networks - Part 1. Adventures in Machine Learning [online]. Available at:
https://adventuresinmachinelearning.com/
improve-neural-networks-part-1/
About Thomas A:
https://adventuresinmachinelearning.com/about/
(Accessed 10th September 2021)

[21] TensorFlow 2 manual: Tensorflow 2 and Keras layers [online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/layers
(Accessed 10th September 2021)

[22] Brownlee J. (2020). Softmax Activation Function with Python. Machine Learning Mastery [online]. Available at:
https://machinelearningmastery.com/softmax-activation-function-with-python/
(Accessed 10th September 2021)

[23] Glorot X and Bengio Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics, PMLR 9, 249-256. Available at:
http://proceedings.mlr.press/v9/glorot10a.html

[24] He K, Zhang X, Res S, Sun J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. Proceedings of the IEEE International Conference on Computer Vision, ICCV, 1026-1034. Available at:
https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html

[25] Tensorflow 2 manual: Initializer that generates an orthogonal matrix [online]. Available at:
https://www.tensorflow.org/api_docs/python/tf/keras/initializers/Orthogonal
(Accessed 11th January 2022)

[26] Wei H, Xiao L, Pennington J. (2020). Provable Benefits of Orthogonal Initialization in Optimizing Deep Linear Networks. International Conference on Learning Representations. Available at:
https://arxiv.org/abs/2001.05992

[27] Thomas A. (2019). An introduction to entropy, cross entropy and the KL divergence in machine learning. Adventures in Machine Learning [online]. Available at:
https://adventuresinmachinelearning.com/cross-entropy-kl-divergence/
About Thomas A:
https://adventuresinmachinelearning.com/about/
(Accessed 10th September 2021)

[28] Brownlee J. (2020). Dropout Regularization in Deep Learning Models With Keras. Machine Learning Mastery [online]. Available at:
https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/
(Accessed 10th September 2021)

[29] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Sakakhutdinov R. (2014). Dropout: A simple way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, vol. 10 (56), 1929-1958. Available at: https://jmlr.org/papers/v15/srivastava14a.html

[30] Kingma DP, Ba J. (2015). Adam: A method for Stochastic Optimization. Published as a conference paper at the 3rd International Conference for Learning Representations (ICLR), San Diego, 2015. Available at: https://arxiv.org/abs/1412.6980

[31] Brownlee J. (2021). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Machine Learning Mastery [online]. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (Accessed 10th September 2021)

[32] Olah C. (2015). Understanding LSTM Networks. colah's blog [online]. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accessed 5th October 2021)

[33] Hochreiter S and Schmidhuber J. (1997). Long Short-Term Memory. Neural Computation, vol. 9 (8), 1735-1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735