NTNU

Norwegian University of
Science and Technology

# Machine Learning for Gesture Recognition with Electromyography

Tony Chau

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Gunnar Tufte, IDI
Co-supervisor: Stefano Nichele, HIOA

Norwegian University of Science and Technology
Department of Computer Science

# Sammendrag

Om lag 70 millioner hørselshemmede personer bruker tegnspråk som sitt førstespråk eller morsmål, men mangelen på et felles språk mellom hørselshemmede og andre gjør den genrelle kommunikasjonen vanskelig. Målet med denne avhandlingen er å undersøke potensialet for å benytte elektromyografi som et verktøy for å bedre den genrelle kommunikasjonen.

Myo armbåndet, utviklet av Thalmic Labs, er en bevegelse- og kontrollenhet som benytter elektromyografiske sensorer, samt gyroskop, akselerometer og magnetometer for å gjenkjenne bevegelser. Denne avhandlingen beskriver utviklingen av et prototype system som benytter Myo armbåndets elektromyografiske sensorer for å oppdage og tolke enkle tegnspråktegn.

Basert på resultatene fra det tidligere arbeidet og det tilhørende rammeverket utviklet for å gjenkjenne og tolke håndbevegelser ved bruk av IMU (inertial måleenhet) og elektromyografisk data fra Myo armbåndet, vil denne avhandlingen fokusere på utvinning av elektromyografiske egenskaper, og bruke maskinlæring for klassifisering.

Denne avhandlingen presenterer et rammeverk for bevegelsesgjenkjenning, og oppnår en nøyaktighet på $85\,\%$ for 10 forskjellige tegnspråk tegn.

# Abstract

About 70 million deaf people use sign language as their first language or mother tongue, but the lack of a common language between the deaf and hearing individuals makes the general communication difficult. This thesis aims to explore the potential of utilizing electromyography to improve the general communication for deaf people.

The Myo armband, developed by Thalmic Labs, is a wearable gesture and motion control device that use a set of electromyographic sensors, combined with a gyroscope, accelerometer and magnetometer, to detect movements and gestures. This thesis presents a development of a prototype-level system that utilize the Myo armband's electromyographic sensors to detect and translate sign language signs to something intelligible for the hearing individuals.

Based on the previous work and the associated framework developed for gesture recognition using the Inertial Measurement Units (IMU) and Electromyography (EMG) sensors from the Myo armband, this thesis focuses on the EMG feature extraction and using machine learning for gestures classification.

This thesis propose a framework for gesture recognition, which achieved an accuracy of $85\,\%$ for 10 different gestures.

# Contents

# List of Figures

# List of Tables

# Acronyms

**ANN** Artificial Neural Network
**ASL** American Sign Language

**CNN** Convolutional Neural Network

**CWT** Continuous Wavelet Transform

**DFF** Deep Feed Forward Network

**DFT** Discret Fourier Transform

**dm** Difference Margin

**DNN** Deep Neural Network

**DTW** Dynamic Time Wrapping

**DWT** Discrete Wavelet Transform

**EMG** Electromyography

**FT** Fourier Transform

**gm** Gesture Margin

**HMM** Hidden Markov Model

**ICWT** Inverse Continuous Wavelet Transform

**IDWT** Inverse Discrete Wavelet Transform

**IMU** Inertial Measurement Units

**ISTFT** Inverse Short Time Fourier Transform

**LSTM** Long Short Term Memory Network

**MAV** Mean Absolute Value

**MSE** Mean Squared Error

**RMS** Root Mean Square

**RNN** Recurrent Neural Network

**STFT** Short Time Fourier Transform

**vt** Value Threshold

**WL** Waveform Length

**WNN** Wavelet Neural Network

**WT** Wavelet Transform

# Glossary

| | |
|---|---|
| **epoch** | is defined as one forward pass and one backward pass of all the training instances. |
| **FANN** | Fast Artificial Neural Network Library, is a free open source neural network library, which implements multilayer artificial neural networks in C. |
| **non-stationary signal** | is the contrast of a stationary signal. |
| **NumPy** | is the fundamental package for scientific computing with Python. |
| **Pewter** | is an open-source project developed for acquisition, analysis and visualization of raw data from the Myo Armband. |
| **processed signal** | is a signal that has been transformed by any mathematical transformations. |
| **PyWavelets** | is a scientific Python module for Wavelet Transform calculations. |
| **raw signal** | is represented in the time-domain. |
| **stationary signal** | is a signal whose frequency content do not change by time. |
| **TensorBoard** | is a suite of web applications for inspecting and understanding the TensorFlow runs and graphs. |
| **TensorFlow** | is an open source software library for machine learning across a range of tasks, and developed by Google |

# Chapter 1

# Introduction

This chapter gives an introduction to this thesis. In addition to the motivation and problem description, a outline of the thesis is provided.

## 1.1. Motivation

People tend to move their hands when they talk, they make gestures. Gesturing is a well-known phenomenon, found across cultures, ages, and work. Gestures are even found in individuals that are blind from birth [22]. Body movement is a powerful medium for non-verbal interaction [7]. If computers were trained to recognize gestures on top of the traditional user interface elements, like text and speech input, it allows the expand for better expressions and new control alternatives. Speech is a very natural way of communicating, but sound may be inappropriate in certain circumstances that require silence, such as police or military infiltrations, or can even be impossible in the case of deaf people [44].

Sign language is a form for human communication based on visual perception. According to World Federation of the Deaf, there are about 70 million deaf people who use sign language as their first language or mother tongue [41]. Deaf and hard-hearing individuals, who learn the sign language from an early age have to learn both the signed and non-signed varieties that co-exist in the society [5]. However, the majority of people in the society do not use a formal sign language, such as the American Sign Language (ASL). Consequently, this creates communication difficulties between deaf people and hearing people.

This creates a demand for a device that is able to interpret sign languages. A such device is not only practical for the sign language communication, but in every context where gesture

based communication is favorable, such as in a military operation or other circumstances where sound based communication is not appropriate.

## 1.2. Problem Description

The purpose of this thesis is to use the Myo armband to detect and interpret gestures, with the intention of improving the communication for people using signs, such as orchestra directors, traffic policemen, football referees, and deaf people. The purpose of this thesis is not to develop a system for practical purpose, but rather explore capabilities and applications of utilizing Electromyography (EMG) for gesture recognition.

The Myo armband is an out-of-the-box gesture recognition device that uses a set of electromyographic sensors, combined with gyroscope, accelerometer and magnetometer to detect motion or recognize gestures. This thesis focuses on analyzing and classifying the EMG data. EMG is an electrodiagnostic medicine technique for evaluating and recording the electrical activity produced by skeletal muscles.

## 1.3. Project Scope and Outline

Based on previous work and the associated framework developed for gesture recognition using the Inertial Measurement Units (IMU) and EMG sensors from the Myo armband, this thesis focuses on EMG feature extraction and applying machine learning for gesture classification, mainly focusing on Neural Networks and Deep Learning techniques.

Chapter 2 introduces the essential background theories for relevant elements and concepts, such as the Myo armband, deep learning and wavelet analysis. The theoretical backgrounds gives a good basis for understanding the underlying elements of the proposed framework. A detailed description of the framework is given in chapter 4. Chapter 3 introduces related work within the field of gesture recognition and control.

Chapter 5 presents the achieved accuracy and performance of the framework, which is analyzed in chapter 6. Chapter 7 propose the overall summary of the thesis, and presents possible future improvements of the framework.

# Chapter 2

# Theoretical Background

This chapter introduces the essential elements and concepts that are relevant to this thesis. The goal of this chapter, is to give a basis for understanding the underlying operations of essential concepts of the framework, such as the neural networks and the wavelet transforms. Introduction to necessary elements, such as the American Sign Language (ASL), EMG, and the Myo armband, is also given in this chapter.

## 2.1. American Sign Language (ASL)

The pre-defined gestures implemented in the framework are taken from the ASL. The list of pre-defined gestures is given in appendix B. Sign language is a form for human communication based on visual perception, and ASL serves as the predominant sign language of deaf in the United States. ASL signs, in addition to the arm gestures, includes important phonemic components such as movement of the face and torso, but this thesis does not take this into consideration. While ASL utilize the movements of both hands, and a mirrored movement have the same meaning, this thesis focuses only on gestures that are using one hand.

## 2.2. Electromyography (EMG)

Electromyography (EMG) is an electrodiagnostic medicine technique to measure muscle responses or electrical activity produced by skeletal muscles. The nerves control the muscles by electrical signals called impulse, these impulses can be measured and analyzed with the help of EMG sensors [61]. There are several techniques to measure EMG signals, but this thesis utilize only the surface EMG. Surface EMG is a technique where electrodes are placed on the skin overlying a muscle to detect nerve impulses.

EMG can be used to sense isometric muscular activity which does not translate into movements. This makes it possible to detect motionless gestures. One of the main difficulties in analyzing the EMG signals, are the noisy characteristics. Compared to other bio-signals, EMG contains complicated types of noises that is caused by the environment. Interfering factors could be inherent equipment noise, electromagnetic radiations, motion artifacts, and the interaction of different tissues. Pre-processing is required to filter out the unwanted noises in EMG [28]. Because surface EMG does not get direct measurement from the motor unit activation, and many factors can affect the signals, these relations are frequently misinterpreted. Although surface EMG can be a useful measure of muscle activation, there are limits to the information that can be extracted from the signals [18].

## 2.3. The Myo Armband

This thesis is based on data given by the Myo armband. An introduction of the Myo armband is given in this section. The Myo armband (figure 2.1), developed by Thalmic Labs, is a wearable gesture and motion control device that uses a set of EMG sensors, combined with IMU sensors, including gyroscope, accelerometer, and magnetometer, to recognize gestures [56].

**Figure 2.1**: The Myo Armband
Source: https://www.myo.com/techspecs [65]

### 2.3.1. Hardware

The Myo armband consist of eight EMG sensors and a nine-axis IMU containing gyroscope, accelerometer and magnetometer. The hardware specification is given in table 2.1.

| | |
|---|---|
| **Sensors** | EMG sensor,<br>Gyroscope,<br>Accelerometer,<br>Magnetometer |
| **Processor** | ARM Cortex M4 Processor |
| **Feedback** | Dual Indicator LEDs,<br>Short, Medium, and Long Vibrations |

**Table 2.1**: List of the Myo armband hardware specifications

## 2.3.2. The Myo SDK

Thalmic Labs provides a SDK that allows developers to obtain the data measured by the Myo armband. The library at the core of the Myo SDK allows applications to interact with the Myo armband. Functionalities in libmyo are exposed through a plain C API. Typically, applications do not interact with the libmyo C API directly, but use a language binding corresponding to the programming language used by the application [57]. Figure 2.2 illustrates the Myo development stack from an application using the SDK down to a physical Myo armband. The Myo armband provides two type of data: Spatial data and gestural data.

### 2.3.2.1. Spatial Data (IMU)

Spatial data represents the position of the armband, and provide information about the orientations and the movements. These data are provided by the IMU. The Myo armband provides IMU data from three different sources:

- Accelerometer
- gyroscope
- and megnetometer

The IMU have a sampling rate of $50\,\mathrm{hz}$. The data from the accelerometer and gyroscope are represented by 3D-vectors, with *g-force* (accelerometer) and *deg/s* (gyroscope) as the units, The orientations from the magnetometer are represented as quaternions. Quaternion representation is an alternative orientation representation to Eulers angles.

### 2.3.2.2. Electromyographic Data (EMG)

The EMG data give information about the orientation independent movements, such as hand gestures. The EMG sensors have a sampling rate of $200\,\mathrm{hz}$. The MyoSDK docu-

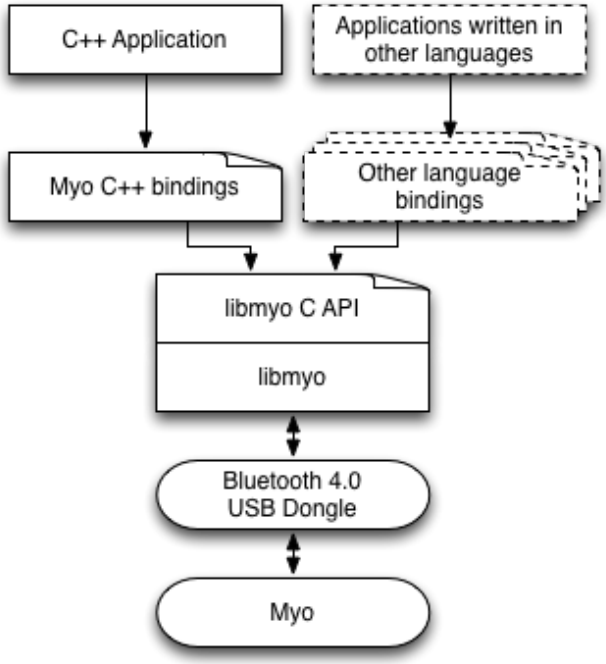



**Figure 2.2**: The Myo SDK stack: Myo development stack from an application using the SDK down to a physical Myo armband.
Source: https://developer.thalmic.com/docs/api_reference/platform/the-sdk.html[66]

mentation [57] does not include information about the unit, but the values are converted into 8-bit values ranging from -128 to 128.

## 2.4. Artificial Neural Network (ANN)

Deep learning and Artificial Neural Network (ANN) are the main classification methods used in this thesis, and this section aims to give a basic understanding of the concepts of deep learning and ANN.

### 2.4.1. Introduction to Artificial Neural Networks

ANNs are computational models used in machine learning, consisting of a large collection of simple interconnected processing units called artificial neurons. A simple ANN is given in figure 2.3. The concept of ANNs are inspired by the biological neural networks and tries

to simulate the biological learning processes. The ANNs are designed to solve problems like a human, by learning from examples, rather than being explicitly programmed.

Like other machine learning methods, ANNs are used to solve a wide variety of tasks within areas that are difficult to solve using ordinary rule-based programming, such as computer vision and speech recognition.

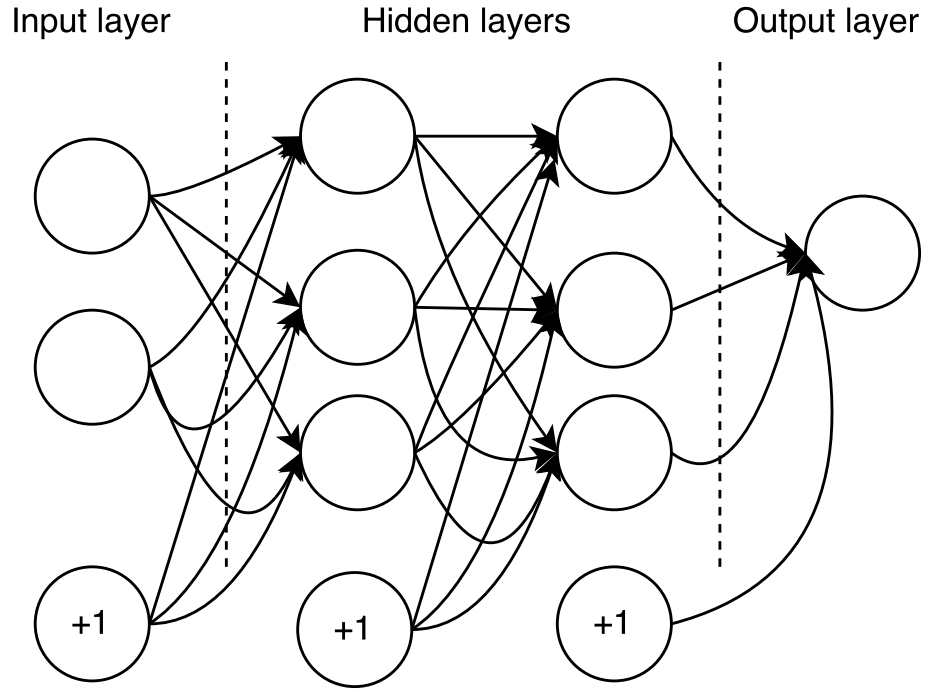



**Figure 2.3**: A simple Feed-Forward Neural Network with two hidden layers. The circles labeled ”+1” are called bias units.

There are many types of ANN, which are described in section 2.4.5, but this chapter mainly focuses on the description and behavior of a multilayer feed-forward neural network with supervised learning.

ANNs are composed of a number of neurons (nodes) with links (synapses) connecting them. The operations of a neuron are given in figure 2.4. Each link have a numeric weight, which are the primary long-term storage of the network, and the learning involves modifying these weights. Figure 2.3 shows that the collection of neurons can be divided into three types of layers:

- The input layer
- The hidden layer(s)
- The output layer

Neurons are connected between layers and signals travel from the input layer, to the output layer. The neurons in the input layer and output layer are connected to the external environment. The weights are calibrated to try to bring the network's output closer to the environment providing the inputs. Some of the neurons, the hidden units, have no direct connection to the external environment, and cannot be directly observed by noting the input and output behavior. A network can have $n$ hidden layers, hence the use of the term Deep Learning. The hidden layers follow a black-box model, and helps the network extract information of something complex, contextual, or non-obvious, such as an image.

In machine learning, there are two types of learning: Supervised and Unsupervised learning. An ANN is said to learn supervised, if the desired output is known. In unsupervised learning, the network learn by input data with no associated output. This paper will only focus on neural network with supervised learning, which consist of two main operations: Forward Propagation and Back Propagation.

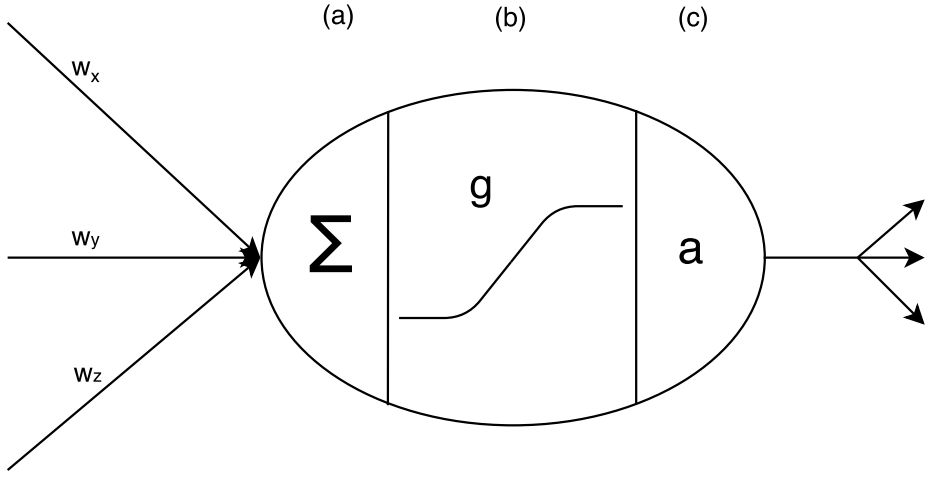



**Figure 2.4**: A diagram illustrating the operation of a Neural Network Unit (neuron). An input goes through an (a) **input function**, which returns a value $\Sigma$. The final neuron output, the activation (c) $a$ is obtained by applying the (b) **activation function** on $\Sigma$, given by $g(\Sigma) = a$.

### 2.4.2. Forward Propagation

In forward propagation, the input travels through the network while getting applied to a set of weights. Each neuron, except for those in the output layer, returns an activation value that travels to the neurons of the next layer. The computation of the activation value is illustrated in figure 2.4. When a neuron receive values from the linked neurons from the

previous layer, the neuron calculates a value $\Sigma$, which is given by

$$\Sigma = \sum_i w_i * a_i \tag{2.1}$$

where $w_i$ and $a_i$ is the weight and the activation from the linked neuron $i$. The activation value $a$ is obtained by applying an activation function $g(x)$ on $\Sigma$, such that

$$a = g(\Sigma) \tag{2.2}$$

The activation function of a neuron defines the activation value given by a set of inputs. There are many types of activation functions, such as linear, sigmoid, ReLu and softmax, but this thesis focuses on the sigmoid activation function, which is defined as

$$f(x) = sigmoid(x) = \frac{1}{1 - e^{-x}} \tag{2.3}$$

The graphical representation of the sigmoid function is given in figure 2.5b. The purpose of the activation functions is presented in section 2.4.4.1. Forward propagation is the operation to compute the output. The output could be used to obtain an error of the network. The error is obtained by comparing the given output with a desired output. To minimize the error, the network propagate backwards by finding the derivative of error with respect to each weight and then subtracting this value from the weight value, the operation is called Back Propagation.

## 2.4.3. Back Propagation

The purpose of back propagation is to calibrate the weight, so that the network causes the output to be closer to the desired output, thus minimizing the error for each output neuron. A cost function is used to calculate the error of the output neurons. A cost function returns a single value $E_{tot}$, refereed to as the total error. More details about the cost function is given in section 2.4.4.2

The back propagation process can be divided into two levels, the output layer and the hidden layer.

### 2.4.3.1. Output Layer

Consider an output neuron $o_x$, connected with a link of weight $w_i$. The goal is to calculate how much a change in $w_i$ affects the total error $E_{tot}$, which can be written as the derivative of $E_{tot}$ with respect to $w_i$,

$$\delta_i = \frac{\partial}{\partial w_i} E_{tot} \tag{2.4}$$

Consider now $a_x$ and $\Sigma_x$ as the activation value and input value of $o_x$. By applying the chain rule, equation (2.4) can be written as

$$\delta_i = \frac{\partial E_{tot}}{\partial a_x} * \frac{\partial a_x}{\partial \Sigma_x} * \frac{\partial \Sigma_x}{\partial w_i} \tag{2.5}$$

where $a_x$ and $\Sigma_x$ are calculated by the activation function and the input function, respectively. To decrease the error, a new weight $\hat{w_i}$ is calculated by subtracting $\delta_i$ from the current weight

$$\hat{w_i} = w_i - \eta\delta_i \tag{2.6}$$

where $\eta$ is the learning rate.

#### 2.4.3.2. Hidden Layers

The operation on the hidden layers is similar to the operation on the output layer, but slightly different since the activation value of each hidden neuron contributes to the activation value of multiple neurons on the next layer.

Consider a hidden neuron $h_y$, connected to a link of weight $w_j$. Let $a_y$ and $\Sigma_y$ denote the activation value and input value, respectively. Given by the same calculations as equation (2.5), the given equation is obtained

$$\delta_j = \frac{\partial E_{tot}}{\partial a_y} * \frac{\partial a_y}{\partial \Sigma_y} * \frac{\partial \Sigma_y}{\partial w_j} \tag{2.7}$$

where $\delta_j$ tells how much a change in $w_j$ affects the $E_{tot}$. Since $a_j$ affects all the activation values of the connected neurons on the next level, $\frac{\partial E_{tot}}{\partial a_y}$ needs to take consideration to its effect on output neurons. Let $L$ denote all neurons receiving input value from neuron $h_y$, then

$$\frac{\partial E_{tot}}{\partial a_y} = \sum_{n \in L} \frac{\partial}{\partial \Sigma_n} E_{tot} \frac{\partial}{\partial a_y} \Sigma_n \tag{2.8}$$

Since

$$\frac{\partial \Sigma_k}{\partial a_y} = w_{yk}$$

where $w_{yk}$ is the weight of the link from neuron $h_y$ to neuron $n_k$. equation (2.8) can be written as

$$\frac{\partial E_{tot}}{\partial a_y} = \sum_{n \in L} \frac{\partial E_{tot}}{\partial a_n} \frac{\partial a_n}{\partial \Sigma_n} w_{yk} \tag{2.9}$$

Note that the original weight of $w_{yk}$ is used, and not the updated weight. The equation (2.9) shows that the derivative with respect to $a_y$ can be calculated if all the derivatives with respect to the activation values $a_n$ of the next layer.

As with equation (2.6), the calculation of the new weight is given by

$$\hat{w_j} = w_j - \eta\delta_j \tag{2.10}$$

where $\eta$ is the learning rate.

## 2.4.4. Neural Network Components

Sections 2.4.2 and 2.4.3 describe the behavior of a multilayer feed-forward neural network with supervised learning. While these sections introduce terms as Activation Function, Cost Function and Bias Units, there are no explanation of these components. This section aims to give a more detailed understanding of the various components in the ANNs.

### 2.4.4.1. Activation Functions

The activation functions determines the activation value of neurons given by a set of inputs, and makes the ANNs non-linear. To understand the importance of the activation functions, consider a fully-connected feed-forward neural network. Let $L_i$ denotes the activation values of the $i$-th layer, and let $i = 0$ be the input layer. Then, by the equation for a fully-connected feed-forward network, $L_i$ is given by

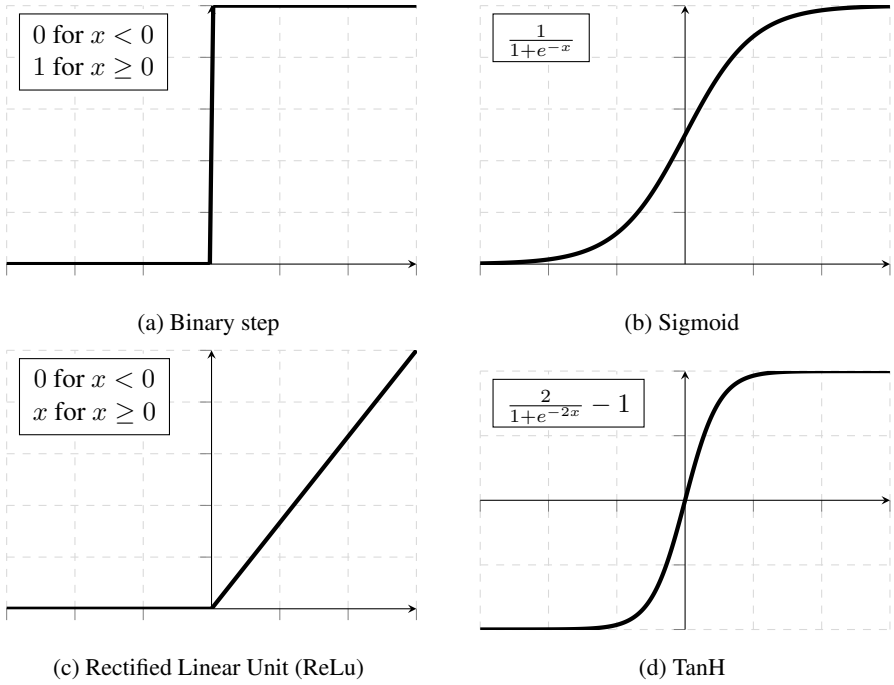
$$L_i = g(W_i L_{i-1}), \qquad \text{for} \quad i \geq 1$$

where $g$ is the activation function and $W_i$ is the weight of the links to the $i$-th layer. If the activation function $g$ is removed, then

$$\begin{aligned} L_i &= W_i L_{i-1} \\ &= W_i W_{i-1} L_{i-2} \\ &= W * L_0 \end{aligned} \tag{2.11}$$

where $W = W_i W_{i-1} \cdots W_1$. Equation (2.11) shows a linear transformation, which is not strong enough to model many kinds of data.

There are many types of activation functions, such as Binary Step, Sigmoid, ReLu, and TanH. The graph representation of the activation functions mentioned are given in figure 2.5. Different activation functions have different properties, and choosing the most optimal activation function is difficult. In [36], a research of choosing the activation function is presented. While a network allow multiple different activation functions, it is common that all neurons in the network use the same activation function [52, p. 567].




(a) Binary step

(b) Sigmoid

(c) Rectified Linear Unit (ReLu)

(d) TanH

**Figure 2.5**: Several Activation Functions

#### 2.4.4.2. Cost Functions

In back propagation, the cost function is used to compute the error of the output layer, and can be defined as

$$C(W, B, X_j, Y_j)$$

where $W$ is the weights, $B$ is the bias units, $X$ is the input of a training instance $j$, and $Y$ is the targeted output of the training instance $j$. There are many different cost functions, and without going into more details, a list of some common cost functions is given below.

- Quadratic cost (Mean Squared Error)
- Cross-entropy cost (Bernoulli negative log-likelihood)
- Exponentional cost
- Hellinger distance
- Kullback–Leibler divergence

There are no generalized rules for choosing the the most optimazed cost function for the problem, and for the simplicity of this thesis, only the Mean Squared Error (MSE) is used.

MSE is defined as

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(\hat{y}_i - y_i)^2 \tag{2.12}$$

where $n$ is the number of output neurons, $\hat{y}_i$ and $y_i$ is the targeted and actual output value of output neuron $i$, respectively.

A cost function must satisfy a few properties:

1. The cost function $C$ must be able to be written as an average

   $$C = \frac{1}{n}\sum_{x} C_x$$

   over a set of cost functions $C_x$ for individual training instances $x$.

2. The cost function $C$ must only be dependent on the activation values of the output layer, since the equation for finding the gradient of the output layer is the only layer that is dependent on the cost function, while the other layers are dependent on the layer on the next level.

3. In order for the gradient descent to work, the cost function $C$ must be differentiable with respect to all the outputs ($y \in Y$).

#### 2.4.4.3. Bias Units

The Bias Units, labeled "+1" in figure 2.3, are neurons that are attached to the end of the input layer and the hidden layers. The bias units do not have any incoming links, and are therefore not affected by the activation values from the previous layer, but they contribute to the activation values of the next layer.

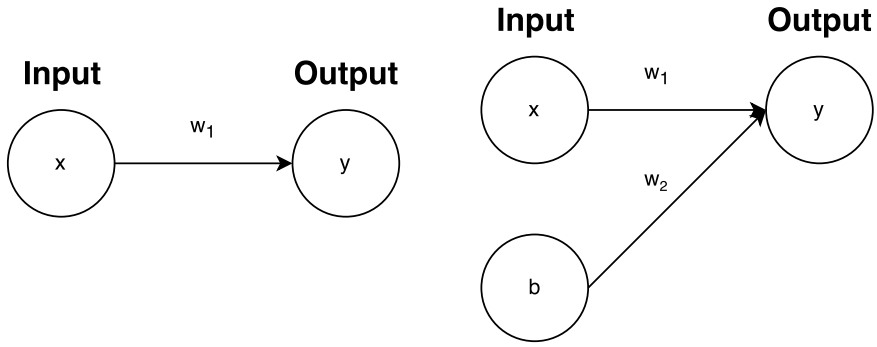
Consider a simple network $G$, illustrated in figure 2.6a, with one input $x$ and one output $y$, connected with a link of weight $w_1$. Let $g(\Sigma)$ be the activation function, then the output $y$ is given by

$$y = g(w_1 x) \tag{2.13}$$

The calibration of $w_1$ will essentially change the steepness of the activation function. The main purpose of the bias unit is to provide every node with a trainable constant value. Adding a bias unit to the simple network $G$, as shown in figure 2.6b, allow the activation function to be shifted to left or right, given by

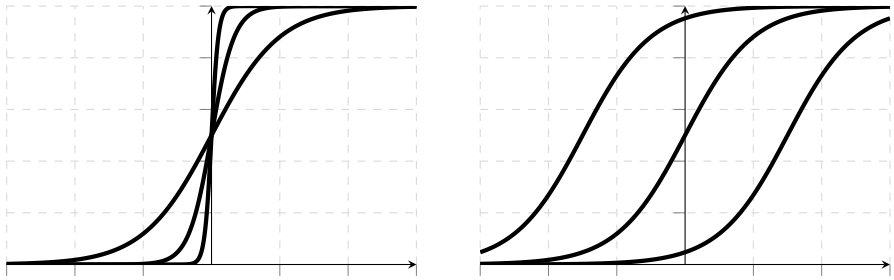
$$y = g(w_1 x + w_2) \tag{2.14}$$

A graphical representation is given in figure 2.7.




(a) (b)

**Figure 2.6**: Two simple Neural Networks with only one input $x$ and one output $y$, where the left network (a) miss the bias unit $b$, while the right network (b) have.



(a) (b)

**Figure 2.7**: The effect of the Bias Unit on the sigmoid function. figure (a), without the bias unit, shows the changes of steepness, while figure (b), with the bias unit, shows the shift of the activation function.

### 2.4.5. Types of Neural Networks

This thesis mainly focuses on the simple fully connected Deep Feed Forward Network (DFF)s. A brief introduction to relevant networks is given in this section. A chart showing characteristic of some network architectures is given in figure 2.8.

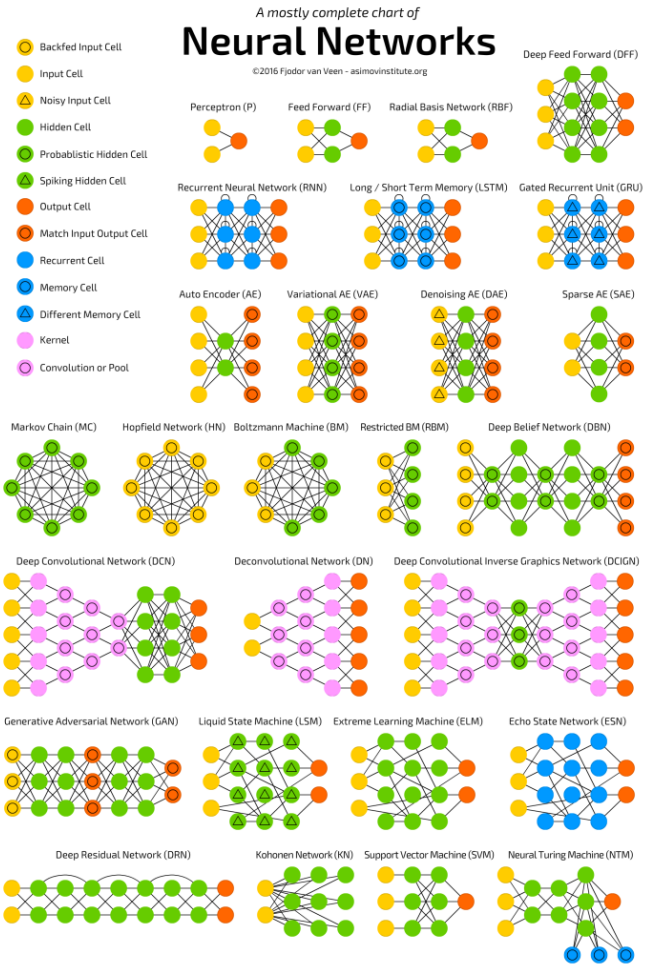



**Figure 2.8**: A chart showing different types of neural network architectures.
Source: http://www.datasciencecentral.com/m/blogpost?id=6448529:BlogPost:470286[64]

#### 2.4.5.1. Deep Neural Network (DNN)

A Deep Neural Network (DNN) is an ANN that contains more than one hidden layer. The DFF is a type of DNN. Each layer trains on a distinct set of features, that are based on the features given by the previous layer. The principle is the further the data advance into the network, the more complex features the network is able to recognize. This is because the deeper layers utilize and combine the features from the previous layer, and is known as hierarchical Feature Learning. Hierarchical feature learning was used before the field of deep learning, but suffered from major issues such as the vanishing gradient problem

where the gradients became too small to provide a learning signal for very deep layers, thus making these architectures perform poorly when compared to shallow learning algorithms [15].

#### 2.4.5.2. Convolutional Neural Networks (CNN)

There are many type of DNNs, and one type which may be relevant for this thesis are the Convolutional Neural Network (CNN). The name "convolutional" indicates that the network employs the mathematical operation, convolution. A convolution is an integral that expresses the amount of overlap of one function $g$ as it is shifted over another function $f$. Convolutional networks are simply ANNs that use convolution in place of general matrix multiplication in at least one of their layers [23, p. 330].

The primary purpose of convolution in CNNs is to extract features from the input. CNNs process data in a grid-like topology, such as time-series data, which can be thought of as an 1D-grid with samples at regular time intervals. The CNNs use a set of smaller "windows", called filters. Each filter is replicated across the input. These replicated units share the same parameterization (weights and bias) and form a feature map. Weight sharing increases learning efficiency by substantially lower the degrees of freedom of the problem. Figure 2.9 illustrate an example of 3 hidden units that belongs to the same feature map with the shared weights $w_1$, $w_2$ and $w_3$. In the back propagation operation, gradient descent can still be used to update the weights, but with a small change to the algorithm. The gradient of a shared weights is simply the sum of the gradients of the parameters being shared. The strength of CNNs is that it can legitimately make stronger assumptions by extracting features from local areas of the input, instead of analyzing the input globally.

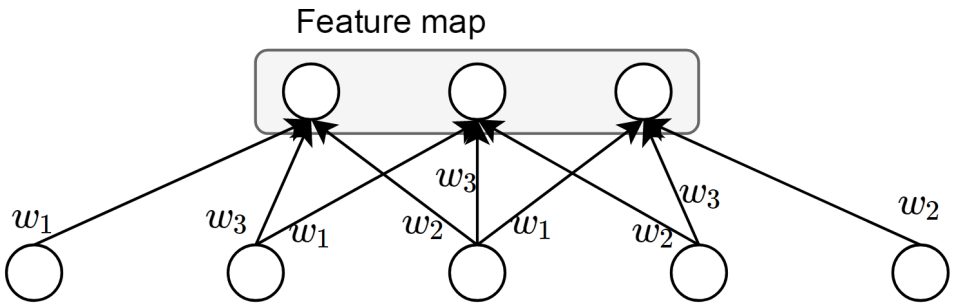



**Figure 2.9**: 3 hidden units that belongs to the same feature map.

#### 2.4.5.3. Recurrent Neural Network (RNN)

Up until now, only networks that make the input move from the input layer to the output layer, the feed-forward networks, have been explained. The feed-forward networks

assume all inputs are independent, and can be categorized as networks without memory. This section gives an introduction to another type of network, the Recurrent Neural Network (RNN). RNNs are a class of ANN where connections between units form a directed cycle.

Unlike the feed-foward networks, RNNs make use of the sequential information. To predict the next word in a sentence, it is helpful to know the previous words. Figure 2.10 illustrates a RNN unrolled. Unrolling means to extract the RNN into a sequense of networks. For example, if the interesting sequence is 5 words, the network would be unrolled into a 5-layer network.

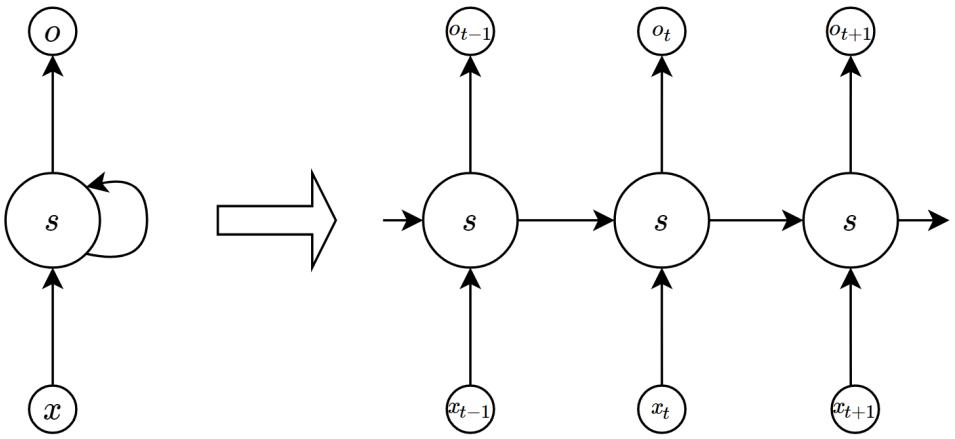



**Figure 2.10**: This figure illustrates a RNN unrolled into a sequence of networks, where $x$ is some input, $s$ is the hidden state, and $o$ is some output.

Theoretically, RNNs should be able to handle context from the beginning of the sequence, which would allow a more accurate predictions of a word at the end of the sequence, but in practice, RNNs are limited to only a few previous steps [21]. The approach is the Long Short Term Memory Network (LSTM)s. LSTMs are a special type of RNNs, which are capable of learning long-term dependencies. In standard RNNs, the repeating module $s$ will have a very simple structure, while LSTMs the repeating module in LSTMs are more complex [43]. This thesis does not explore the behavior of the LSTMs, but it is worth mentioning that the LSTM units includes a memory cell that can maintain information in the memory for a long period of time.

### 2.4.6. TensorFlow

The framework use the TensorFlow library developed by the Google Brain Team. Google Brain Team is Google's machine intelligence team that focuses on deep learning. TensorFlow provides a Python API, as well as C++, Haskell, Java, Go, and Rust APIs.

TensorFlow is an open source software library for machine learning, that use computational graphs for numerical computations [55]. A computational or dataflow graph can be thought of as a form of directed graph, where nodes describe operations and edges represent the data flowing between these operations. The major advantage of representing an algorithm with a computational graph is the ability to show a visual expression of dependencies between units of a computational model, such as the TensorBoard graph, illustrated in figure 2.11. TensorBoard is a suite of web applications for inspecting and understanding the TensorFlow runs and graphs.

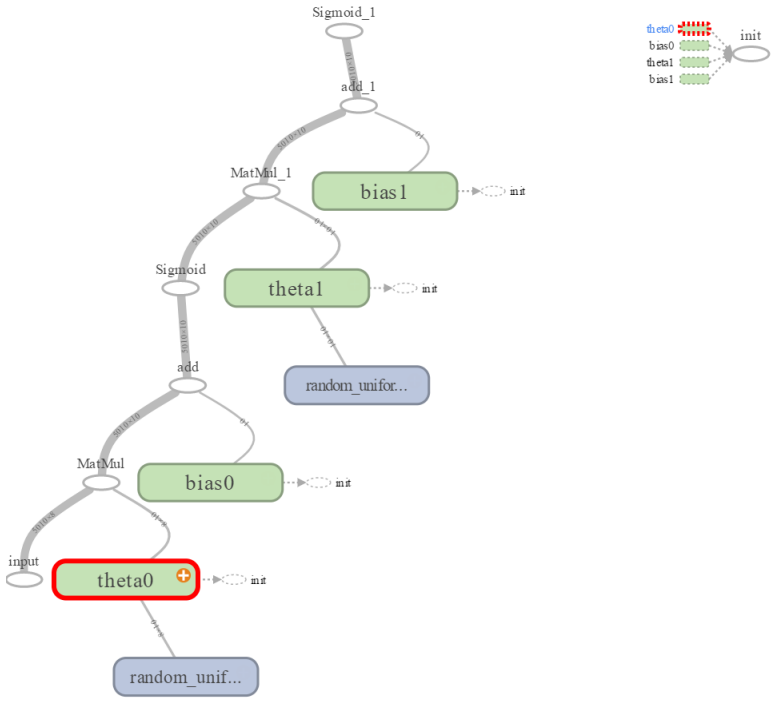



**Figure 2.11**: A TensorBoard graph showing a simple fully conneted neural network, with 3 layers. The big nodes labeled $theta0$ and $theta0$ refers to the hidden and output layer, respectively. The smaller nodes in the graph are the operation nodes.

In TensorFlow, edges represent data flowing from node to node, and are referred to as tensors. Mathematically, a tensor is the generalization of a N-dimensional matrix. In terms of computational graphs, a tensor can be seen as a connection between operations. A tensor does not hold or store values in the memory, but provides an interface for retrieving the values referenced by the tensor.

Python was the first client language supported by TensorFlow, and the Python API is currently the API that supports the most features [55]. There is a performance loss of Python compared to C++, which is discussed in section 4.2. However, TensorFlow's core backend is implementation using C++, which makes it compiler optimized. It is also worth mentioning that the Python API integrates well with NumPy, a package for scientific computing. The TensorFlow graph runs in sessions, where the API specify which operations to execute in the run-function. Outside data may be supplied to placeholders so the graph can run multiple times with different inputs.

TensorFlow supports both CPU and GPU devices. This is an important note, because back propagation, can be stated as a set of matrix multiplications. The performance of CPU and GPU version is discussed in section 6.1.1.1.

## 2.5. Signal Processing

EMG signal processing is a vital part of this thesis, and this section introduces the signal processing technique used, the Wavelet Transform (WT).

In signal processing, mathematical transformations are applied to signals to obtain information that is not readily available in the raw signals. This section uses the notation raw signals for the time-domain signals, and processed signals for the signals that have been transformed by any mathematical transformations.

Usually, measurable signals are time-domain signals, in other words, a signal given by a function of time. In many cases, distinguished information can be hidden in the frequency content of the signal. When the time-domain signal is plotted, a time-amplitude representation is obtained. The time-amplitude representation have one of the axes representing the time, and the other the amplitude. In comparison, the frequency-amplitude representation have one of the axes representing the frequency and the other being the amplitude. The frequency-amplitude representation gives information of how much each frequency exist in our signal. To obtain the frequency-amplitude representation, an transformations, such as the Fourier Transform (FT), can be applied on the signal. Since the WT was developed as an alternative to Short Time Fourier Transform (STFT), and to overcome some resolution related problems coming with the STFT [47], the FT was mentioned specifically. Signals represented in the frequency domain allow the signals to be manipulated on a different perspective, in areas such as reducing noise, compress data, modulate, filter and encode.

### 2.5.1. The Fourier Transform

To get a better understanding of the WT, an introduction to Fourier Transform (FT), which is a necessary background, is provided. Only essential theory is provided, since this subject is fairly wide.

The FT decomposes a signal from the time-amplitude representation into the frequency-amplitude representation, an illustration is given in figure 2.12. FT is based on the principle that any signal can be represented by an equation consisting of a combination of $\sin$ and $\cos$ functions.

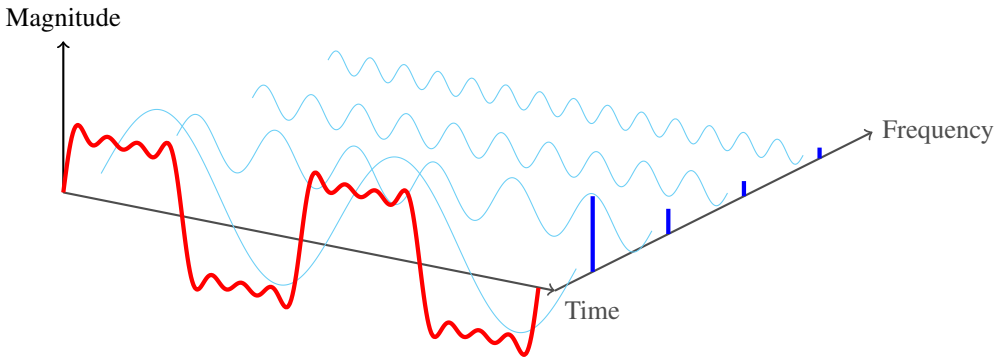



**Figure 2.12**: A visual representation of the FT.

The FT of a signal $f(t)$, is defined as

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(t)\, e^{it\xi}\, \mathrm{dt} \tag{2.15}$$

for any real number $\xi$. The FT is a reversible transform, meaning it has the property to go back and forward between the raw signal and processed signal. The Fourier Transform (FT) is given by

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{it\xi}\, \mathrm{d}\xi \tag{2.16}$$

An important note are the limits $-\infty$ and $\infty$, meaning the signal $f(t)$ get integrated over all time. This is an important note that is explained later.

There are two type of signals, stationary signals and non-stationary signals. Signals whose frequency content do not change by time are called stationary signals, and in contrast the non-stationary signals change the frequency content over time. Consider a stationary signal $f_1(t)$, shown in figure 2.13a, and given by

$$\begin{aligned} f_1(t) = {} & cos(2\pi \cdot 10t) + cos(2\pi \cdot 25t) \\ & + cos(2\pi \cdot 50t) + cos(2\pi \cdot 100t) \end{aligned} \tag{2.17}$$

and a non-stationary signal $f_2(t)$, shown in figure 2.13b, and given by

$$f_2(t) = \begin{cases} sin(2\pi \cdot 100t) & \text{if } 0 \leq t < 0.2 \\ sin(2\pi \cdot 50t) & \text{if } 0.2 \leq t < 0.5 \\ sin(2\pi \cdot 25t) & \text{if } 0.5 \leq t < 0.8 \\ sin(2\pi \cdot 10t) & \text{if } 0.8 \leq t \leq 1 \end{cases} \tag{2.18}$$

where $t$ is $0 \leq t \leq 1$ and given in *seconds*. In $f_1(t)$ the frequencies $10$, $25$, $50$ and $100$ exist at all time, while in $f_2(t)$ the same frequencies exist, but not simultaneously. By applying the FT, such that

$$f_1(t) \xrightarrow{\mathscr{F}} \hat{f}_1(\xi) \tag{2.19}$$

$$f_2(t) \xrightarrow{\mathscr{F}} \hat{f}_2(\xi) \tag{2.20}$$

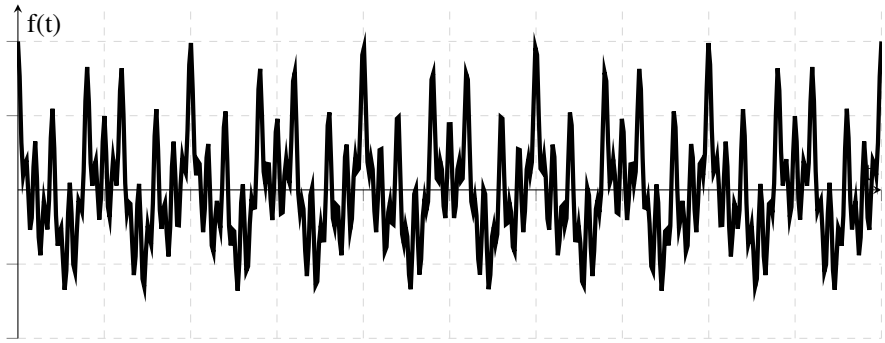
The obtained signals $\hat{f}_1(\xi)$ and $\hat{f}_2(\xi)$ are given in figure 2.14. While we can see that the figure 2.14a and 2.14b are not entirely the same, we can see that both graphs have four major peaks at the same frequencies values, that are $10$, $25$, $50$ and $100$ hz. The noise in $\hat{f}_2(\xi)$, that is representet as the small ripples in figure 2.14b, is due to sudden changes from one frequency component to another. The difference of the amplitudes of the frequencies is due to the duration of the frequencies. The point of this example is to show the close similarity of two processed signals, even though the raw signals are quite different.

The FT makes the frequency information readily available, but in exchange readily available information about time. The FT is quite useful on analyzing stationary signal, but because of the loss of time information, it may not always be as useful on non-stationary signal . Unfortunately, most of the natural signals are non-stationary, such as the electrical activity of the body (electrocardiograph, electroencephalograph and electromyography). To approach those signals it is more useful to use a revised version of the FT that gives information of both time and frequencies, the STFT.

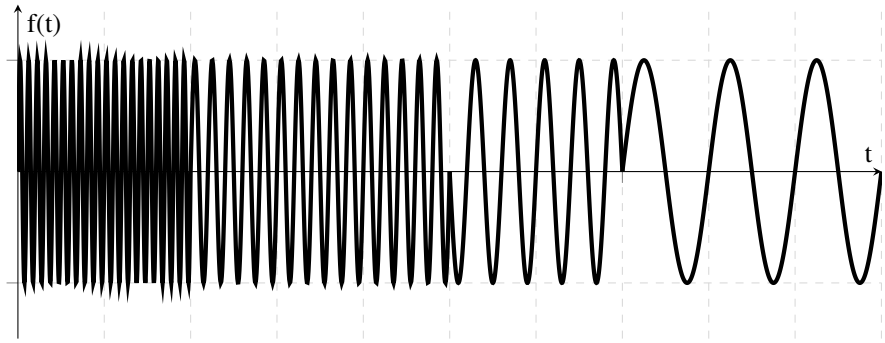
### 2.5.2. The Short-Time Fourier Transform and The Uncertainty Principle

The FT is a common method to extract frequency-information from a signal, but it is not always useful when time plays an essential part of the signal analysis. Short Time Fourier Transform (STFT) is an approach on the FT that introduces some time dependence into the FT, a sliding window function $w(t-u)$. Mathematically this can be written as

$$\hat{f}_s(u,\xi) = \int_{-\infty}^{\infty} f(t)\, w(t-u)\, e^{-it\xi}\, \mathrm{dt} \tag{2.21}$$




(a) A stationary signal with frequencies of 10, 25, 50, and 100




(b) A non-stationary signal with frequencies of 10, 25, 50, and 100, on different times.

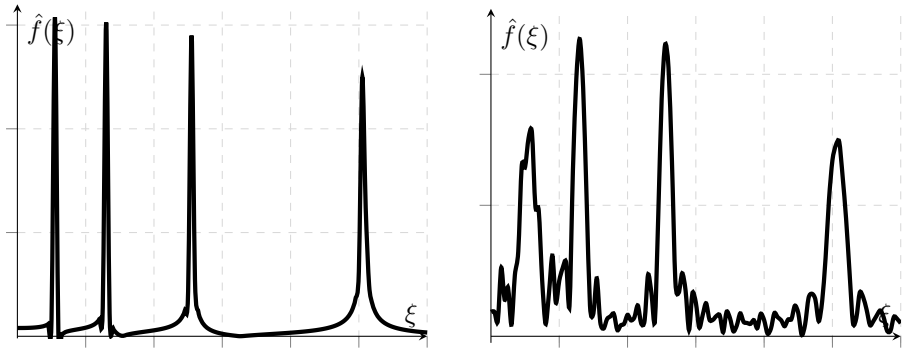
**Figure 2.13**: A stationary signal and a non-stationary signal are given in figure (a) and (b), respectively. Both graphs contain the frequencies 10, 25, 50 and 100.

Like the FT, the STFT is also reversible. The Inverse Short Time Fourier Transform (ISTFT) is given by

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}_s(u, \xi)\ w(t-u)\ e^{ix\xi}\ \mathrm{d}\xi\ \mathrm{du} \tag{2.22}$$

Instead of processing the whole signal, STFT divide the raw signal into shorter segments of equal length, called windows. Then applies the FT separately on each segment. The STFT gives a time-frequency representation as shown in the spectrograms given in figure 2.15. Figures 2.15a and 2.15b are the spectrogram of the signals given by the equations (2.17) and (2.18), respectively. The spectrograms give information about which frequencies appear at a given time interval.

The limitation is referred to as Gabor limit. It is related to Heisenberg's Uncertainty Principle. In psychics, the uncertainty principle says that we cannot measure both the position and the momentum of a particle with absolute precision. In other words, the more accu-




(a) The Fourier Transform of the stationary signal in figure 2.13a.

(b) The Fourier Transform of the non-stationary signal in figure 2.13b.

**Figure 2.14**: The FT of a stationary signal and a non-stationary signal is given in figure (a) and (b), respectively. Both graphs have major peaks at the frequency of 10, 25, 50 and 100.

rately one value is known, the less accurately the other value is known. In signal processing, the Gabor limit is about the time–frequency resolution limit. If we define the standard deviations $\sigma_t$ of the time and standard deviations $\sigma_f$ frequency, then we can formally write Gabor limit as

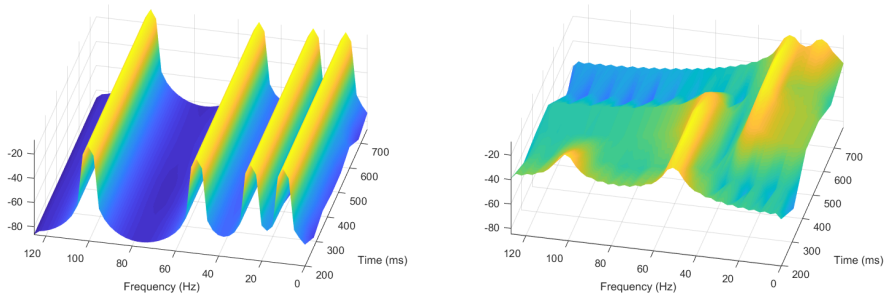
$$\sigma_f \sigma_t \geq \frac{1}{4\pi} \tag{2.23}$$

The width of the windows determine whether there is good frequency resolution or good time resolution, and the limit of STFT is the fixed resolution.

A wide window gives better a frequency resolution but a poor time resolution. A narrower window gives good a time resolution but poor frequency resolution. The problem is choosing an optimal window function for the analysis. If the frequency components are well separated in the raw signal, then one may sacrifice some frequency resolution and choose a good time resolution, since the spectral components already are well separated. However, if this is not the case, then a good window function could be more difficult than finding a good stock to invest in [47]. WT seems to be an approach to the limit problem of STFT [12].

### 2.5.3. Wavelet Transform

The FT transforms a signal from the time domain to the frequency domain by decomposing the raw signal into a formula consisting of $\sin$ and $\cos$ functions. In comparison, the Wavelet Transform (WT) uses the wavelets. Unlike $\sin$ and $\cos$ waves, that are infinite, the wavelets are limited waves that decreas back to zero. The Continuous Wavelet Transform (CWT) is obtained by convolving a signal with an infinite number of wavelet functions,




(a) (b)

**Figure 2.15**: Figure (a) and (b) shows the spectrogram of the STFT with the same window size of the signal represented in figure 2.13, respectively.

generated by translating ($\tau$) and scaling ($s$) a certain mother wavelet function.

#### 2.5.3.1. Definition

The mother wavelet function is defined as $\psi(t) \in L^2(R)$, which is limited in the time domain. That is, $\psi(t)$ has values in a certain range and zeros elsewhere. The basic wavelets $\psi_{s,\tau}(t)$ are generated from the mother wavelet $\psi(t)$, given by

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi(\frac{t-\tau}{s}) \tag{2.24}$$

The CWT is formally written as

$$\begin{aligned}\gamma(s,\tau) &= \int_{-\infty}^{\infty} f(t)\psi^*_{s,\tau}(t)\ dt \\ &= \frac{1}{\sqrt{s}}\int_{-\infty}^{\infty} f(t)\psi^*(\frac{t-\tau}{s})\ dt\end{aligned} \tag{2.25}$$

where $*$ denotes complex conjugation. The variables $s$ and $\tau$ are the new dimensions after the WT, scale and translation, respectively. Equation (2.25) shows how the signal $f(t)$ is decomposed into a set of basic wavelets $\psi_{s,\tau}(t)$, and maps an one-dimensional signal to a two dimensional coefficients $\gamma(s,\tau)$. The transform makes it possible to locate a particular frequency $s$ at a certain time instant $\tau$.

As with FT and STFT, the WT have an inverse. If the signal $f(t)$ is a $L^2(R)$ function, the Inverse Continuous Wavelet Transform (ICWT) can formally be written as

$$f(t) = \frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty}\gamma(s,\tau)\ \frac{1}{\sqrt{s}}\ \psi(\frac{t-\tau}{s})\ \mathrm{d}\tau\ \frac{\mathrm{d}s}{s^2} \tag{2.26}$$

where $C_\psi$ is defined as

$$C_\psi = \int_0^\infty \frac{|\Psi(\xi)|^2}{\xi} \, \mathrm{d}\xi < \infty \tag{2.27}$$

$\Psi(\xi)$ is the Fourier transform of the mother wavelet $\psi(t)$. That is,

$$\psi(t) \xrightarrow{\mathscr{F}} \Psi(\xi)$$

Equation (2.27) is the admissibility condition. The most important properties of the wavelets are the admissibility and the regularity conditions [60]. The admissibility condition makes the wavelet a wave and the regularity condition gives it the fast decay.

### 2.5.4. Discrete Wavelet Transform

Most signals are given by a discrete representation, and smooth varying parameters is not always necessarily to obtain interesting signal features or reconstruct the signal from the wavelet coefficients. For this kind of signals, the Discrete Wavelet Transform (DWT) may be sufficient. The main concept of the DWT is the same as the CWT, but with a significant reduction of computation time. Unlike the Discret Fourier Transform (DFT), which is a discrete version of the FT, the DWT is not really a discrete version of the CWT.

The CWT is computed by continuously shifting a continuously scalable function over a signal and calculating the correlation between those two. In DWT, different cutoff frequency filters are used to analyze the signal at different scales. The signal is passed through a series of high-pass filters, the mother wavelet, to analyze the high frequencies. Then the singal is passed through a series of low-pass filters, the father wavelet, to analyze the low frequencies.

The basic DWT passes the raw signal $x$ through a half-band digital low-pass filter with impulse response $g$. A half-band low pass filter removes all frequencies that are above half of the highest frequencies. Filtering a signal corresponds to the mathematical operation of convolution of the signal with the impulse response of the filter. The convolution operation in discrete time is defined as

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{2.28}$$

The signal is decomposed simultaneously using a high-pass filter $h$. As illustrated in the diagram given in figure 2.16, the filters return the detail coefficients (the high-pass filter) and approximation coefficients (the low-pass). The detail coefficients and approximation coefficients is then subsampled by the factor of 2. Subsampling by a factor $n$ is to reduce the number of samples in a signal by $n$ times. Since half of the frequencies of the signal have been removed from the cutoff frequency filters, half the samples can be discarded according to Nyquist's rule, which says the sampling's frequency must not be above half the sampling frequency [6]. On this point the scale of the signal have been doubled.

Even though low-pass filtering removes the high frequency information, the scale remain unchanged. By the subsampling process the scale get changed. Resolution is related to the amount of information in the signal, and it is affected by the filtering operations. Filtering half of the frequencies with a half-band low-pass filter can be interpreted as losing half of the information, and this will affect the resolution. However, the subsampling operation after filtering does not affect the resolution, since removing half of the spectral components from the signal makes half the number of samples redundant.

Let the subsampling operator be defined by $\downarrow$, given by

$$(y \downarrow k) = y[kn] \tag{2.29}$$

Then mathematically, the process can be express as follow

$$y_{low}[k] = (x * g) \downarrow 2 \tag{2.30}$$
$$y_{high}[k] = (x * h) \downarrow 2 \tag{2.31}$$

where $y_{low}[k]$ and $y_{high}[k]$ are the outputs of the low-pass and high-pass filters after subsampling by the factor of 2, respectively.

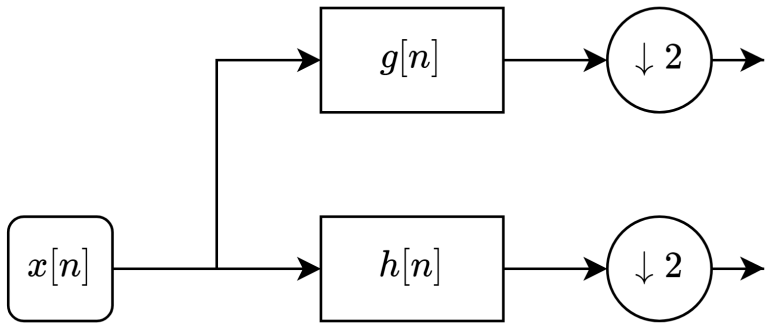



**Figure 2.16**: Wavelet decomposition at level 1. We apply a low-pass filter $g$ and a high-pass filter $h$ to a signal $x$, then subsample the output by the factor of 2.

The described decomposition is repeated for further decompositions. On each level the approximation coefficients from the low-pass filter is decomposed with high and low-pass filters and then subsampled. At each level, the filtering and subsampling will result in halve the time resolution and double the frequency resolution.

The DWT can be interpreted as computing the wavelet coefficients of a discrete set of child wavelets for a given mother wavelet $\psi(t)$. The mother wavelet is shifted and scaled by powers of two, and by a little modification of the equation (2.24) we get

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi(\frac{t - k2^j}{2^j}) \tag{2.32}$$

DWT is used to decompose signals, but can be assembled back into the original signal without loss of information. This process is refereed to as reconstruction, and is done by the the Inverse Discrete Wavelet Transform (IDWT). Wavelet decomposition involves filtering and downsampling, while the wavelet reconstruction consists of upsampling and filtering. The reconstruction is shown in figure 2.17. Upsampling is the opposite of downsampling, the process of lengthening a signal component by inserting zeros between samples.

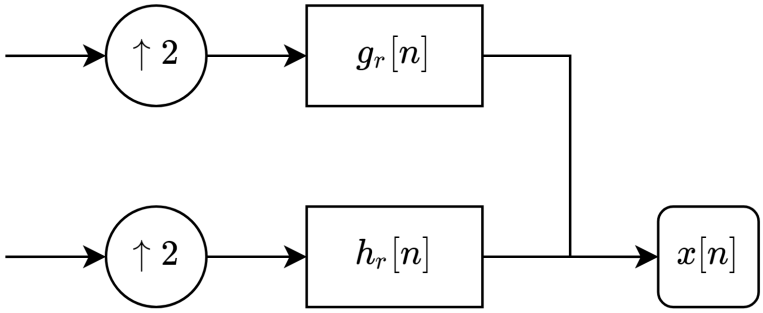



**Figure 2.17**: A wavelet reconstruction diagram, where $g_r$ and $h_r$ are the low reconstruction and high reconstruction filters, respectively.

## 2.5.5. Comparison of Wavelet and Short-Time Fourier Transform

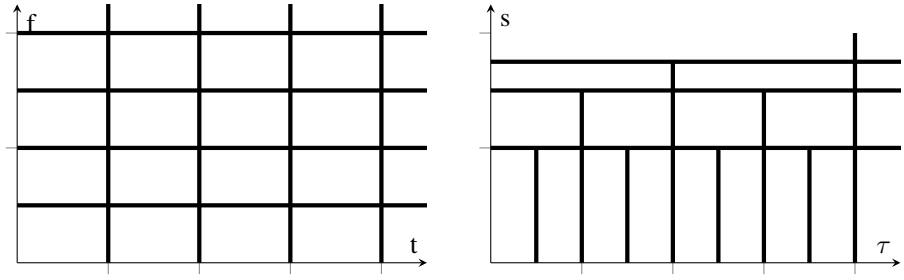
WT and STFT are both developed to approach the problem with time information by FT. This section present some differences and similarities of the features of WT and STFT.

### 2.5.5.1. Resolution

In STFT, the flexibility is restricted by that one fixed size of the time window is selected for all the frequencies. The limit of flexibility causes problems when the signal requires more accurate information for the other element. The WT approach this by a windowing technique with variable-sized regions. The ability to changes the time extension in wavelet transform, allow the use of

- long time intervals where more precise low frequency information is required and
- shorter time intervals where high-frequency information is required.

A comparison of the resolution of STFT and WT is shown in figure 2.18, where each box represents an equal portion of the time-frequency plane. Note that the area for both the WT and the STFT resolution are constant, determined by the uncertainty principle.




(a) Resolution of short-time Fourier Transform (b) Resolution of wavelet transform

**Figure 2.18**: Comparison of (a) STFT and (b) WT resolution.

#### 2.5.5.2. Scale and Translation

Unlike the STFT, WT dose not give a time-frequency representation by time and frequency, rather by the translation $\tau$ and scale $s$.

The term translation is strictly related to time, and corresponds to time information in the transform domain. Translation gives information of where the mother wavelet is located. The translation of the mother wavelet can be thought of as the time elapsed since $t = 0$.

The WT does not have the frequency parameter, instead the scale parameter is used. The scale is inverse of frequency. That is, high scales correspond to low frequencies, and low scales correspond to high frequencies. The relationship between the wavelet scale and the equivalent Fourier frequency depends on the wavelet and the sampling frequency.

#### 2.5.5.3. PyWavelets

The framework is implemented in Python, and uses PyWavelets for the wavelet analysis. PyWavelets seem to be the only well-developed library for the wavelet calculation for Python. PyWavelets is a free and Open Source wavelet transform software for the Python programming language [49].

Some main features of PyWavelets are:

- $1D$, $2D$ and $nD$ DWT and IDWT
- 1D CWT
- Computing Approximations of wavelet and scaling functions
- Over 100 built-in wavelet filters and support for custom wavelets
- Single and double precision calculations

- Real and complex calculations

The most important relevant features are DWT and IDWT, and the built-in wavelet filters.

# Chapter 3

# Background

This chapter present some related work for gesture control and recognition. Section 3.2 covers the machine learning relevant related work, while section 3.1 covers the generalized case.

## 3.1. Related Work

In this section, work related to gesture recognition is presented. This includes both research on gesture controlled devices and gesture recognition.

### 3.1.1. EMG-based Controlled Robots

Various interface systems and prosthetics have been developed to support handicapped people with limited manipulation capability of the upper limb due to traffic accidents, cerebral apoplexy, or other afflictions. Many prosthetic arms have been developed for amputees since the 1970's, and in [19], the concept of an EMG-based human-robot interface as rehabilitation aids is proposed. In [3], a methodology for controlling an anthropomorphic robot arm using nine surface EMG electrodes to record the muscular activities is proposed. A control interface is proposed, according to which, the user performs motions with the user's upper limb. The recorded electromyographic activity of the muscles can be transformed into kinematic variables that are used to control an anthropomorphic robot arm.

In [13], a project to help amputee Johnny Matheny that lost his arm to cancer in 2008 is presented. At the Johns Hopkins Applied Physics Laboratory, Matheny worked with a prosthetic arm attached directly to his skeleton, this prosthetic arm is controlled by the

use of two Myo armbands on the upper arm that detects the electrical activity of the muscles.

### 3.1.2. Gesture Recognition

The study of human-computer interaction has developed to put a significant amount of effort on user-friendly interfaces employing voice, vision, gesture, and other innovative I/O channels. One of the most challenging approaches in this research field is to link neural signals with computers by exploiting the electrical nature of the human nervous system. The development of an EMG-based interface for hand gesture recognition is presented in [28], where an EMG-sensor is positioned on the inside of the forearm to recognize control signs in the gestures.

Gesture-based control is one of the major application for hand gesture recognition technologies. Another major application is sign language recognition. Sign Language recognition aim to help the deaf communicate with the hearing society conveniently. In [63], a framework for hand gesture recognition based on the information of a three-axis accelerometer and multichannel EMG sensors is presented.

### 3.1.3. Alternative Gesture Control Methods

The ability of tracking position and movement for gesture recognition can be achieved by various approaches. The Myo armband uses accelerometer, gyroscope, and magnetometer to keep track of the orientation, and EMG-sensors to measure electric activity of the muscle. Other tools such as Kinect and Leap Motion is based on a more visual approach.

#### 3.1.3.1. Gesture Gloves

Hand gestures are movements of the arms and fingers, and one possible approach to track movement of the arms and fingers is to place sensors directly on the fingers, for example in the form of a glove. Two University of Washington undergraduates have won a $10,000 Lemelson-MIT Student Prize for gloves that can translate sign language into text or speech [42]. Another example is the out-of-the-self gesture glove from Maestro Gesture Glove [20].

#### 3.1.3.2. Image-based Gesture Control

Leap Motion [31] and Kinect [29] are two commonly used sensors for tracking motions by utilizing technology such as infrared cameras, infrared LEDs, RGB camera, and depth sensor. Unlike the other mentioned devices, Leap Motion and Kinect observe the movements, instead of touching the user. Leap Motion and Kinect utilize different technologies,

thus have different advantages and disadvantages, but conceptually, they work in the same manner. There are some different ASL related work based on both Leap Motion [48, 11] and Kinect [62, 30, 8], but also work that tries to utilize the advantage of both by combining the sensors [35].

### 3.1.4. SCEPTRE

This thesis was mainly inspired by a project SCEPTRE by some researchers from Arizona State University [44]. The primary goal of SCEPTRE is to match gestures.

The system SCEPTRE is comprised of an Android smartphone or a Bluetooth enabled computer and one to two Myo devices. The goal is to develop a system using two Myo devices to decipher ASL gestures, and display the meaning on a smartphone or computer. SCEPTRE utilizes the data from the accelerator, magnetometer, and EMG-sensors. The project is an attempt to develop a system toward a system which is ubiquitous, non-invasive, works in real-time, and can be trained interactively by the user.

The system is envisioned to be used in two primary applications:

- User-to-User interaction
- User-to-Computer interaction

20 ASL signs with training instances for each gesture were chosen to test the system. It is also possible for the user to train the system with additional signs, either in **guided mode** or **ASL mode**. In guided mode the system compares the new gesture data with the existing data collection to ensure there are no clashes, meaning too much overlapping data. Unlike guided mode, the ASL mode does not make this guarantee.

Dynamic Time Wrapping (DTW) is used to compare accelerometer and orientation data. For the EMG data, an energy based comparison was used. By using a combination of data from the accelerometer, magnetometer and EMG sensors the system was able to achieve an accuracy of $97\,\%$.

## 3.2. Machine Learning

This section introduces related work that utilize machine learning technique for gesture recognition. Section 3.2.1 focuses on work using ANNs and deep learning, while section 3.2.2 focuses on work using other machine learning techniques such as the Hidden Markov Model (HMM).

### 3.2.1. Neural Network

There are many researches on classification of EMG signals using ANN. An example can be found in [54, 25, 4]. The method presented in [54] use a Wavelet Neural Network (WNN) to classify neuromuscular disorders. WNN is described as a class of networks that combine the classic ANN and the wavelet analysis [2]. Unlike the use of wavelet analysis in this thesis, the discrete wavelet function is used as the neuron's activation function, and not in feature extraction for the inputs. In [25], a prosthesis is designed to be controlled by EMG signals detected by surface electrodes attached to the user's arm. A similar work is introduced in [4], where CNNs are used for classification of movements for prosthetic hands

### 3.2.2. Alternative Machine Learning Techniques

HMM are a tool for modelling time series data, which dominate speech recognition [26]. Speech recognition have some relations with gesture recognition. In [32], an EMG based speech recognition system is introduced, and [9] demonstrate that myoelectric signal automatic speech recognition using an HMM classifier is resilient to temporal variance. The HMM used for continuous gesture recognition is presented in [33, 16, 10]. Note that these works are not based EMG data, but on orientation and trajectories.

# Chapter 4

# Methodology

This chapter introduces the concepts and methods used to develope the framework. The framework is based on the previous work, and section 4.1 provide a description and achievements of this preliminary work.

## 4.1. Previous Work

The core of the current framework was developed in the preliminary work of this thesis. This section uses the term preliminary framework to refer to framework developed in the previous work and current framework to the framework developed for this thesis. The preliminary system consisted of a Myo armband, connected via Bluetooth to a computer. The preliminary framework was implemented using C++, because the C++ bindings was included in the Myo SDK.

Cross Correlation and Dynamic Time Wrapping (DTW) are the two methods used in the preliminary framework to classify gestures. Cross correlation and Dynamic Time Wrapping (DTW) are two different methods to measure the similarity of two graphs. Classification of a gesture was determined by the similarity of the data of the unknown gesture to the data from a set of known gestures. Cross correlation is a method to estimate the degree of similarity of two series. Dynamic Time Wrapping (DTW) is a technique to find an optimal alignment between two given time-dependent sequences. The goal is to align two sequences of graph by warping the time axis iteratively until an optimal match between the two graphs is found.

Unlike the current framework, the preliminary framework utilized all data provided by the Myo armband. The scope of this thesis is only to analyze the data given by the EMG senors, thus the data given by the IMU sensors are ignored. The preliminary framework using cross correlation and Dynamic Time Wrapping (DTW) achieved an accuracy around

90% by utilizing both IMU and EMG data. However, removing the analysis of the EMG data improved the results slightly. Removing the analysis of the IMU data, on the other hand gave results without any clear characteristics. From the results of the previous work, one can assume with high certainty that cross correlation and Dynamic Time Wrapping (DTW) are not practical to extract information from the EMG signals. It is worth mentioning that the test for these results was fairly small, with only 10 instances of 5 different gestures.

## 4.2. System Description

The system developed with this thesis consists of a Myo armband, connected via Bluetooth to a computer. The outer-overview of the system is shown in figure 4.1. The computer uses Myo Connect, drivers provided by Thalmic, to receive data from the Myo armband. By using the MyoSDK, described in section 2.3.2, the framework gets an easy access to the sampled data. Unlike the preliminary framework, described in section 4.1, the current framework is implemented in Python instead of C++. The Myo development stack diagram is given in figure 2.2.

Unlike C++, which is a compiled language, Python is an interpreted language. Compiled languages are known to have a better run time than interpreted languages, because of optimization achieved by directly using the native code of the target machine. The tests used on the preliminary work, with cross correlation, was approximately four time faster on the C++ implantation, than on the equivalent Python implementation. The higher level of abstraction is the main reason for the choice to transit to a Python implementation. Python has a strong position in scientific computing [27]. Open source libraries, such as NumPy, give Python a high performance for numerical calculations. Numerical calculations are an essential part of the EMG feature extraction procedure.

This thesis focuses mainly on ANN and EMG feature extraction techniques, and Python have support for a number of well developed deep learning and signal processing technique libraries, and therefore seems like a acceptable choice for this thesis. While TensorFlow, the machine learning library used in the framework, support a C++ interface, the Python API seem more developed at the moment, and comes with a more comprehensive documentation.

The source code of the current framework is published in the git repository given in appendix A.1.

## 4.3. Neural Network

The framework use ANN to classify gestures based on the EMG signals obtained from the Myo armband. The theory of the ANN is explained in section 2.4, and this section describes the practical use of ANN in the framework.

(a)

(b)

**Figure 4.1**: Figure (a) shows a user wearing the Myo armband performing a gesture. Figure (b) shows the Myo armband connected to the system on a computer.

There are many open source libraries developed for machine learning, and specifically for ANNs. As mentioned in section 4.2, the early version of the framework was implemented in C++. The early phase of framework used an ANN library called FANN, Fast Artificial Neural Network Library [40]. FANN is a free open source ANN library providing features for a simple ANN. While FANN provides a simple implementation, it is limited to simple networks. The leading libraries for deep learning seem to be TensorFlow[55] and Theano[58]. Both libraries mainly implemented for Python. There are no concrete reasons for the system to use TensorFlow over Theano, but TensorFlow seem to be better marketed. Although, FANN provide enough features to produce the networks for the current framework, TensorFlow opens the possibility for future development with more complex networks such as the RNNs, introduced in section 2.4.5.3. A brief performance review of FANN and TensorFlow will be presented in chapter 5.

### 4.3.1. Architeture

There exist many type of ANN architectures as shown in figure 2.8. However, this thesis focuses only on the DFF architecture. While the chart given in figure 2.8 shows the characteristics of many different types of network, there are still underlying structures of network.

The network architecture used in this thesis are the fully connected multi-layer DFF described in section 2.4. In this types of network there are some parameters to determine the structure,

- the number of input neurons,
- the number of output neurons,

- the depth of the network,
- and the number of hidden neurons on each hidden layer.

The number of input neurons in the input layer $\theta_0$ is determined by the EMG features, more details about the EMG features are given in section 4.4. The number of output neurons in the output layer $\theta_n$ is determined by the number of gestures the network can recognize. Each gesture is linked with an output neuron. The link is determined by the position of the neuron in the output and the gesture ID. When we apply some input into the network, each output neuron will return some value. This values range from 0 to 1, and the closer the value is to 1, the higher the possibility is for the input to be the data from the gesture that the output neuron is linked with.

The depth of the network and the number of hidden neurons is each layer are two parameters with more uncertainty. If a network is too big, it can memorize all the examples by forming a large lookup table, but may not be the most optimal for inputs that the network have not seen before. In other words, ANN are subjected to overfitting when there are too many parameters. It is known that a feed-forward network with one hidden layer can approximate any continuous function of the inputs, and a network with two hidden layers can approximate any function at all [52].

This thesis experiments with different values for the mentioned parameters, and examines if there are any significant change on the results. The main structure used in this thesis is illustrated in figure 4.2. The number of hidden neurons on each hidden layer $\theta_i$ is determined by the size of the input layer $\theta_0$, but does not necessarily have to be the same. The determination of size of the hidden layer is given by a simple rule that the size of the hidden layer has to be greater than the size of the input layer, so size$(\theta_i) >$ size$(\theta_0)$, where $i \in [1, n-1]$. This thesis also experiments with the depth $n$ of the network. For the simplicity of experimenting with different value of $n$, the number of neurons in each hidden layer $\theta_i$ is the same.

### 4.3.2. The Implementation

The framework is implemented using TensorFlow, introduced in section 2.4.6. The framework provides a simple method to create networks of different depth and customizable size for each hidden layer. It also provide a simple method to construct the network with different activation functions for each layer, but this thesis mainly focuses on the sigmoid activation function. On the other hand, the system gives no opportunity to change the cost function without editing the source code, and only MSE is used.

When the system creates a new network, it creates the associated training file and network metadata file. The training file gives information about the number of training instances, the input size, the output size, and most importantly, it contains the pre-calculated input values. More information regarding the input is presented in section 4.4.3.

The network metadata file contains details about the network, such as

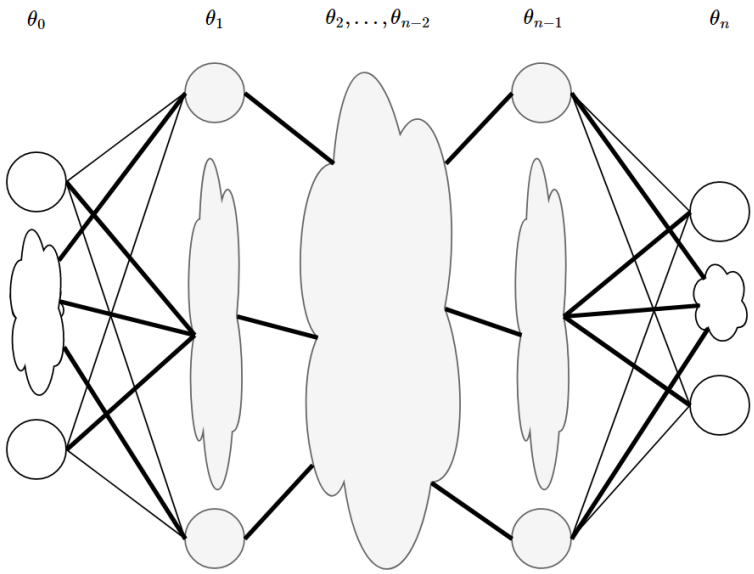



**Figure 4.2**: A diagram showing the structure of the networks used in this thesis. $\theta_i$ for $i \in [1, n-1]$ are the hidden layers, while $\theta_0$ and $\theta_n$ are the input layer and output layer, respectively. The clouds represent a set of neurons, and the heights of the clouds indicate the size of layer.

- layer sizes,
- epoch count,
- activation function for each layer,
- wavelet level,
- and which features are used.

The layer sizes are represented by an array, and includes information about size of the input and output, and also gives information about the depth of the network by the length of the array. Each element in the array represent the the number of neurons (excluding the bias units) in the layer level corresponding to the element's index. An epoch is defined as one forward and one backward propagation of all the training instances, in other word, a forward and backward propagation of all training instances given in the training file. The information about the wavelet level and the features makes it easier for the system to analyze the unprocessed data with the right features. The information of which features are used is represented by an array, with the value $0$ or $1$. $1$ means the feature is included, while $0$ means the feature is excluded.

The information about the structure of the different networks given in appendix E is based on the information given in the network metadata file.

## 4.4. EMG Feature Extraction

EMG signals have the properties of non-stationary, nonlinear, complexity, and large variation, which makes it difficult to analyze and classify. To build a system based on EMG, we have to extract some information from the signals. There are various approaches and methods for EMG feature extraction, such as the Wavelet Analysis, Auto Regressive Analysis, Spectral Magnitude Averages and Fourier Analysis [53].

EMG feature extraction is a technique to extract essential and usable information that may be hidden in the EMG signal, in addition to remove unnecessary parts and interference. The feature analyze of EMG signals can be divided into three main groups [46]:

- Time domain,
- frequency domain, and
- time-frequency domain

This thesis focuses on feature extraction from the time domain and time-frequency domain. This thesis choose to use the wavelet analysis, since the WT is a time-frequency analysis method that is designed for analyzing non-stationary signals such as the EMG signals, and seems to be an effective tool to extract useful information from the EMG signals [45].

### 4.4.1. Wavelet Transform

WT can process signals that are non-stationary and time varying in nature. A technical description of WT is given in section 2.5, and this section covers the basis of how wavelets analysis are used in the framework.

There are two types of WT methods, the DWT and CWT. As explained in sections 2.5.3 and 2.5.4, DWT have a significant reduction in the computation time compared to CWT, and therefor the DWT is chosen as main supplement for the EMG feature extraction.

There exist many types of wavelets, but there is no generalized method to choose the mother wavelet [39]. The Daubechies Wavelets is chosen because it is one of the most common wavelet family. Daubechies wavelets are a family of orthogonal wavelets that are characterized by a maximal number of vanishing moments. Daubechies wavelets is refered to as db$N$. Each wavelet has a number of vanishing moments equal to the number $N$ of coefficients. For simplicity, this thesis only uses db1.

If an EMG signal $S$, pass through a low-pass filter and a high-pass filter, we obtain an approximation coefficient subset $cA$ and a detail coefficient subset $cD$. A diagram of the progress is given in figure 4.3. The coefficients of filter depends on wavelet function type. The depth of the decomposition tree is determined by the wavelet decomposition level (wavelet level).

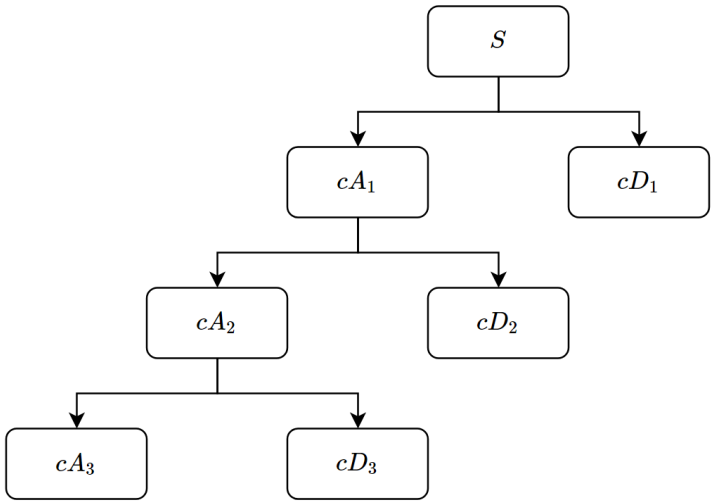



**Figure 4.3**: Wavelet decomposition of level 3. We apply a low-pass filter and a high-pass filter to the EMG signal $S$, to obtain an approximation coefficient subset $cA$ and a detail coefficient subset $cD$.

Consider a decomposition of level $n$, then from the wavelet decomposition we obtain the wavelet coefficient subsets

$$cD_1, \ldots, cD_n, \quad \text{and} \quad cA_n$$

Each coefficient subset can be reconstructed to obtain an effective EMG signal part. Reconstruction of a signal is done by using the IDWT, explained in section 2.5.4. Generally, the IDWT is performed by using the final-level approximation coefficient $cA_n$ and all the detail coefficients $cD_i$. However, this thesis define the reconstruction of the EMG signal by the inversion of subset dependence. The reconstructed EMG signals, namely

$$D_1, \ldots, D_n, \quad \text{and} \quad A_n$$

are reconstructed from wavelet coefficient subsets $cD_1, \ldots, cD_n, cA_n$, respectively. For example the reconstructed EMG signal $A_n$ is reconstructed by using the coefficients of the final-level approximation coefficient $cA_n$ only.

The obtained wavelet coefficient subsets and the reconstructed EMG signals are further used in the feature analysis that is explained in section 4.4.2.

### 4.4.2. Features

There are thirty-seven features presented in [46], where the majority are defined in time domain, and only a minority are defined for the frequency domain. Many of the presented features share similarities, and it is therefore not necessary to use all features. For this thesis, using [34] as the basis for the choice, the Mean Absolute Value (MAV), Root Mean Square (RMS), and Waveform Length (WL) is used. The mathematical operations for MAV, RMS, and WL is given in sections 4.4.2.1 to 4.4.2.3. The diagram in figure 4.4 shows the procedure of the EMG feature extraction from the wavelet decomposition and reconstruction of the EMG signal.

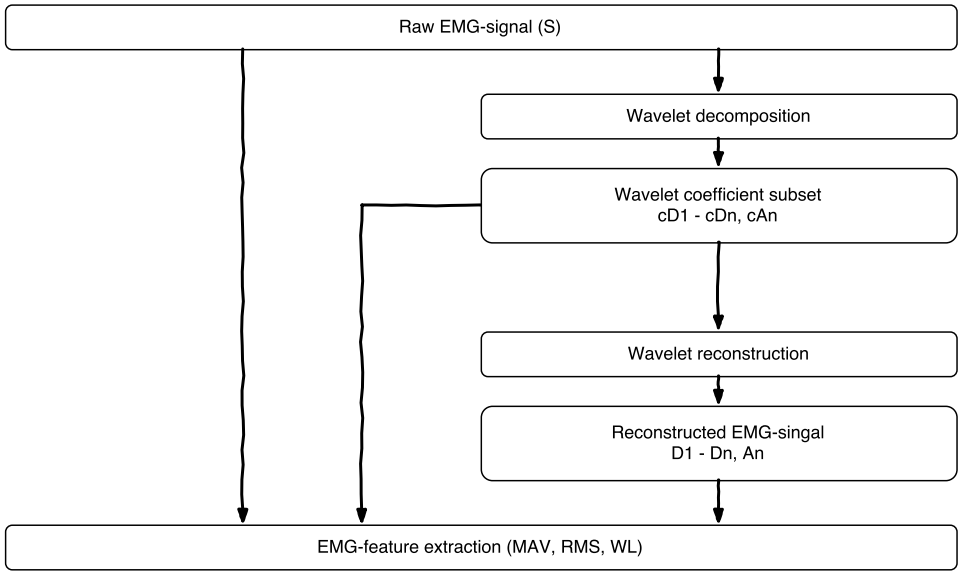



**Figure 4.4**: The EMG feature extraction procedure from an EMG signal

#### 4.4.2.1. Mean Absolute Value

Mean Absolute Value (MAV) is one of the most popular feature in EMG signal analysis [46]. MAV is the average of the absolute values of the EMG signal amplitudes, and is defined as

$$MAV = \frac{1}{N}\sum_{n=1}^{N}|x_n| \tag{4.1}$$

#### 4.4.2.2. Root Mean Square

Root Mean Square (RMS) is also another popular feature in EMG signal analysis [46]. It is modeled as amplitude modulated Gaussian random process whose relates to constant force and non-fatiguing contraction. The mathematical definition of RMS can be expressed as

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n)^2} \tag{4.2}$$

#### 4.4.2.3. Waveform Length

Waveform Length (WL) measure the complexity of the EMG signal, and we define it as the cumulative length of the EMG waveform over the time segment. Mathematically given by

$$WL = \sum_{n=1}^{N}|x_{n+1} - x_n| \tag{4.3}$$

### 4.4.3. Network Inputs

The size of the input layer is determined by the features, which is described in sections 4.4.1 and 4.4.2. The wavelet level defines the depth of the decomposition, which determines the number of wavelet coefficient subsets and reconstructed EMG signals. An example of a wavelet decomposition of level 3 is given in figure 4.3.

Consider a wavelet decomposition of level $n$, this decomposition will give $n$ detail coefficient subsets and $1$ approximation coefficient subsets. As described in section 4.4.1, the reconstructed EMG signals are reconstructed from each of the wavelet coefficient subsets, giving a total of $n + 1$ reconstructed EMG signals. In addition, the raw EMG signal $S$ is used in the analysis. The Myo armband have in total $8$ EMG sensors. Let the number of features be denoted by $k$, then the number of input $I$ is given by

$$I = ((2(n + 1) + 1) \cdot 8) \cdot k$$

The implemented structure of the input is given in appendix D.2.

The values of of the input are normalized by using NumPy's normalization function $\mathrm{numpy.linalg.norm}$, given by

$$a_n(a) = \frac{a}{\mathrm{numpy.linalg.norm}(a)} \tag{4.4}$$

where $a_n$ is the normalized array of $a$. Normalization is done in groups of eight, the grouping is given in the input structure described in appendix D.2.

The Normalization by grouping seems to have a bug on the input, creating duplicate values. Given the $\mathrm{input}$ function described in appendix D.2, if the the $\mathrm{isRecosntructed}$ parameter is the only different parameter, then the normalization will return the same value. Such that

```
a1 = [input(0, n, FALSE, k),..., input(8, n, FALSE, k)]
a2 = [input(0, n, TRUE, k),..., input(8, n, TRUE, k)]
```

by the normalization given in equation (4.4) give

$$a_n(a1) = a_n(a2)$$

This is further discussed in section 6.1.2.6.

## 4.5. Datasets

This thesis deals with two independent types of dataset. Two type of dataset were necessary because the ANNs needs a sufficient amount of data for the training phase. The dataset is refereed to as the hackathon dataset and the recorded dataset.

### 4.5.1. Hackathon Dataset

The hackathon dataset is a dataset received from the author of [17]. The author claims he obtained the data from a hackathon he participated, and cannot give any guarantee for the data. It is stated in the report that the data files was provided by Thalmic Labs, but there are still no garantee of the data is simulated or sampled.

The dataset is given as a collection of CSV files, given by the following name

$$\text{Gesture}M\text{\_Example}N\text{.CSV}$$

where $M$ is the gesture ID and $N$ is the example ID. The original received dataset used *txt* filetype instead of CSV, but this has no effect on the data. The dataset includes $6$ different gestures, with $2000$ instances of each gesture. It is stated in the actual hackathon challenge task that each file contains a $50\times8$-matrix representing the EMG signals, but this statement is not true. Some files contains a matrix with the row length from $40$ to $49$.

In section 2.3.2.2, it was mentioned that the EMG sensors returned values ranging from $-128$ to $128$. It is not stated that the values are sampled from the Myo armband, but it is worth noting that the hackathon dataset values range from $0$ to $2218$.

### 4.5.2. Recorded Dataset

The Recorded Dataset is the dataset recorded by the framework, and unlike the hackathon dataset, it is control on the data in the recorded dataset. The data is obtained directly from

the Myo API, and stored in JSON-files. The whole dataset is provided in the git repository given in appendix A.2.

It is worth mentioning, that the networks created in the early phase of the framework includes training data from a small deprecated dataset that was not sampled by the framework. From the preliminary framework and the early phase of the current framework, the data was obtained from Pewter. Pewter is an open-source project developed for acquisition, analysis and visualization of raw data from the Myo Armband [14]. Pewter worked for creating a small dataset, but due to ANNs necessity of a large training set, a more suitable method was implemented.

The framework, samples signals from the Myo armband with a time interval. The time interval is set to $4.5\,\text{seconds}$, starting from the user presses the *enter*-button. $4.5\,\text{seconds}$ gives EMG data arrays of size $900$ and IMU data arrays of size $225$, the sampling rate of the EMG sensors and IMU sensors are $200\,\text{hz}$ and $50\,\text{hz}$, respectively. The raw sampling files include a margin. Each gesture takes $1$ to $2\,\text{seconds}$ to perform, and the raw files includes unnecessary data of the gestures being held in their final positions. To reduce the computing time, the framework provides a method to remove the unnecessary data at the end. As for this thesis, the framework uses $3\,\text{seconds}$ of the data. While this method reduce the unnecessary data at the end, the unnecessary data of when the user presses the *enter*-button before doing the actual gesture is still used in the analyzes.

The method for creating gesture data files uses one of the pre-trained networks. The network is trained from earlier data to try to recognize the right gesture for the given data. This method is implemented to optimize the production of training data. The framework allows the user to correct the classification when the system recognizes the wrong gesture.

The recorded dataset files are given by the following name

$$\text{recorded-}X\text{-}N\text{.json}$$

where $X$ is the gesture name and $N$ is the example id. The framework is implemented with $10$ pre-defined gestures given in appendix B. The structure of the json-files is given in appendix D.1. Although the framework does not utilize the IMU data, it is still stored and may be used for future work.

The framework also allows sampling of data with an user id. These data are used to evaluate the framework for a more generalized case. These files are given by the following name

$$I\text{-NotMe-}X\text{-}N\text{.json}$$

where $I$ is the user id, $X$ is the gesture name and $N$ is the example id. In this thesis, these files is refereed to as the ”NotMe” dataset.

# Chapter 5

# Results

This chapter provides the results achieved by the framework. The results include both the performance and accuracy results, given in sections 5.3 and 5.4, respectively. The requirements and conditions for the analyzes is given in section 5.1. The results are discussed more thoroughly in chapter 6.

## 5.1. Result Background

This section gives information about the requirements and conditions for the analysis, in other word the background of the results.

## 5.2. Gesturs

The framework is implemented with 10 pre-defined gestures given in appendix B. Table B.1 provides a lsit of pre-defined gesture with the gesture names and their corresponding gesture id. Since there is no assurance from a qualified sign language user, it is uncertain whether the gesture is correct or wrong in terms of meaning. However, this should not affect the results, as long as the gesture are consistent. The gestures are imitated movement from the ASL dictionary [59]. The gestures are randomly chosen from common words with consideration of signs that mainly use one hand, the other hand may have a supporting role. All the data is sampled from the Myo armband on the right hand.

As described in section 4.5, this thesis uses two different types of dataset. There is no information about the gestures represented in the hackathon dataset, but in this thesis, the gesture name is used to refer to the gesture with the corresponding id for a better readability.

### 5.2.1. Hardware Specifications

The framework has been tested on two different computers, referred to as desktop and laptop. The information about the specifications of the computers is given in appendix C. The specifications is important, because the preformance analysis is dependent on the hardware. In this thesis, processes that run on the laptop only utilize the CPU. When TensorFlow is used on the desktop, the GPU version is used, unless otherwise specified.

### 5.2.2. Training

In order for an ANN to achieve satisfying accuracy of gesture classification, it need a sufficient amount of training data. The hackathon data set consist of $2000$ files of each gesture. $1500$ of files are used for training, while the other are used for testing. The recorded dataset consist of $601$ files, and $501$ are used for training. The "NotMe" dataset, the dataset sampled from other users, range from $24$ to $37$ files for each gesture. The "NotMe" dataset is not used to train any network, and the purpose is to experiment if networks, trained from only one user, are able recognize the gestures from another user.

This thesis gives analysis of the training time, and briefly discuss the quantity of sufficient training. The earlier networks was trained with less than $150$ instances for each gesture. The analysis compares the network that used a small training set with the network that used a significantly bigger training set.

Unless otherwise specified, the training set is of size $9000$ and $5010$ for the hackathon dataset and recorded dataset, respectively.

### 5.2.3. Testing

To compare accuracy of the network structures, the tests are based on a unseen dataset. In the hackathon dataset the test consist of $500$ instances for each gesture, and the in the recorded dataset the test consist of $100$ instances for each gesture.

Since there is limited information about the hackathon dataset, the files with the highest example id is chosen to be the test samples, meaning, example id $N$ from $1501$ to $2000$. For the recorded dataset, the test files are sampled with some parameters in mind. Parameters such as the time of the day, mood, and place.

### 5.2.4. Network Structure Notation

This thesis represents the networks in a table format based on the information from the network metadata files described in section 4.3.2.

While not all created networks are used, information of all the created networks is given in appendix E. Table 5.1 shows an example of a table containing the network information.

$xN_i$ is the network reference, where $x$ indicate which dataset the network is based on. $x = r$ indicates the recorded dataset and $x = h$ indicates the hackathon dataset.

| $rN_0$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [120, 150, 24, 10] |
| Epoch count | 4970000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

**Table 5.1**: A example of a table that is used to represent a network. It is based on the information from the network metadata files.

## 5.3. Performance

This section focuses on the performance of the framework. Unless otherwise stated, the GPU version of TensorFlow is used.

### 5.3.1. Training

For even the relatively small networks used in this thesis, the training took several days. This section purpose the training performance. The analysis includes changes in parameters, such as Network Depth $n$, Number of Inputs $I$, or Size of Hidden Layers $h$, and see if it has any effect on the training time. The run time is calculated by the average of 5 runs on 5000 epochs.

#### 5.3.1.1. Network Depth

Given a network structure with $n$ hidden layers, given by

$$\text{Layer Sizes} \quad [120, h_1, \ldots, h_n, 6]$$

where $h_1, \ldots, h_n = 150$. Table 5.2 shows the training time comparison of run time of 5000 epochs for different values of $n$.

| $n$ | Run time 5000 epochs |
|---|---|
| 1 | 3 m 12 s |
| 2 | 3 m 12 s |
| 3 | 3 m 13 s |
| 4 | 3 m 14 s |

**Table 5.2**: Training time comparison of network with different depth.

#### 5.3.1.2. Number of Inputs

Let the input size be denoted by $I$. Given a network structure

Layer Sizes [$I$, 300, 300, 6]

Table 5.3 shows the training time comparison of run time of 5000 epochs for different values of $I$.

| I | Run time 5000 epochs |
|---|---|
| 24 | 1 m 30 s |
| 120 | 3 m 25 s |
| 168 | 4 m 14 s |
| 216 | 5 m 2 s |

**Table 5.3**: Training time comparison of network with different input size

#### 5.3.1.3. Size of Hidden Layers

This section presented the performance results for changing the size of the hidden layers. For simplicity, all the hidden layers are given the same size. Given a network structure

Layer Sizes [120, $h$, $h$, 6]

Table 5.4 shows the training comparison of run time of 5000 epochs for different values of $h$.

| $h$ | Run time 5000 epochs |
|---|---|
| 150 | 3 m 12 s |
| 200 | 3 m 10 s |
| 300 | 3 m 25 s |
| 600 | 3 m 50 s |

**Table 5.4**: Training time comparison of network with different hidden layer size $h$

### 5.3.2. The Forward Propagation

The run time of the the forward propagation includes EMG feature extraction computation, such as the wavelet decomposition and reconstruction. Unlike the results of the training performance, the forward propagation dose not seem to be noticeably affected by any changes in the network structure. Each gesture is analyzed by approximately 0.1 seconds

## 5.4. Accuracy

Section 5.3 presented results of the performance, this section covers the accuracy. In this thesis, accuracy is used as a mesurment of how good the network is to recognize the correct gesture. This thesis has analyzed three type of data, the hackathon dataset and recorded dataset (and ”NotMe”-dataset). Note that the recorded dataset consists of two type of dataset. The ”NotMe”-dataset is the dataset with data sampled from various unique users. The details about the test sets are given in section 5.2.3.

The framework supports two types of analyzes, raw analysis and analysis with parameters. The result of the raw analysis is presented in section 5.4.1, and the analysis with parameters is presented in section 5.4.2. Note that the network training is dependent on the number of epochs, and not the error rate of the training. This means the network finishes the training when the epoch count reaches a given number.

### 5.4.1. Raw Results

This section present the raw results. Raw results means that the system classify the gestures by choosing the output neuron with the highest value.

Table 5.5 shows the output from a forward propagation in network $rN_5$ with inputs extracted from the data file *recorded-YES-320.json*. With the raw result analysis, the network categorize this *YES* gesture as a *SLEEP* gesture, even though all the values are quite far from 1 and the output values for *SLEEP* and *YES* is relatively close.

| File: recorded-YES-320.json | |
|---|---|
| EAT | 0.019150 |
| HELP | 0.000259 |
| SLEEP | 0.176043 |
| THANKYOU | 0.000256 |
| WHY | 0.000270 |
| NO | 0.015150 |
| YES | 0.157366 |
| DRINK | 0.003311 |
| HELLO | 0.000002 |
| SORRY | 0.001016 |

**Table 5.5**: The output of a forward propagation of network $rN_5$. The file consist data of a *YES* gesture, but the network recognized it as a *SLEEP* gesture in the raw result analysis.

#### 5.4.1.1. Network Depth

Consider the networks $hN_9$, $hN_8$, $hN_5$, and $hN_7$, the networks with layer sizes given by

$$\text{Layer Sizes} \quad [120, h_1, \dots, h_n, 6]$$

where $n$ is the number of hidden layers and $h_i = 150$ for $i \in [1, n]$. Table 5.6 shows the comparison of the mentioned networks with different values of $n$.

| $n$ | Accuracy [%] |
|---|---|
| 1 | 78.53 |
| 2 | 76.50 |
| 3 | 76.43 |
| 4 | 76.97 |

**Table 5.6**: Raw results of the networks with different number of hidden layers $n$, namely $hN_9$, $hN_8$, $hN_5$, and $hN_7$.

Consider also the networks $rN_7$ and $rN_5$ given by

$$\text{Layer Sizes} \quad [216, h_1, \dots, h_n, 10]$$

with $h_i = 300$ for $i \in [1, n]$. Table 5.7 shows the comparison for $rN_7$ and $rN_5$ with different values of $n$.

| $n$ | Accuracy [%] |
|---|---|
| 1 | 95.80 |
| 2 | 96.60 |

**Table 5.7**: Raw results of the networks with different number of hidden layers $n$, namely $rN_7$ and $rN_5$.

#### 5.4.1.2. Wavelet Level

Consider the networks $hN_{11}$, $hN_8$, and $hN_6$, with wavelet level 0, 1, and 3, respectively. Table 5.8 shows the comparison of the results for the networks using the hackathon dataset. For the recorded dataset networks, $rN_4$, $rN_3$, $rN_6$, and $rN_5$, with wavelet level 0, 1, 2, and 3, respectively, the results are shown in table 5.9.

| Wavelet Level | Accuracy [%] |
|---|---|
| 0 | 78.70 |
| 1 | 76.50 |
| 3 | 77.33 |

**Table 5.8**: Raw results of the networks of different wavelet levels, namely $hN_{11}$, $hN_8$, and $hN_6$.

| Wavelet Level | Accuracy [%] |
|---|---|
| 0 | 90.30 |
| 1 | 94.80 |
| 2 | 95.10 |
| 3 | 96.60 |

**Table 5.9**: Raw results of the networks of different wavelet levels, namely $rN_4$, $rN_3$, $rN_6$, and $rN_5$.

#### 5.4.1.3. Features

This thesis uses three EMG features, MAV, RMS, and WL. This section presents the comparison of networks that use different combination of these features. Consider the networks $rN_{10}$, $rN_{14}$, and $rN_5$. Table 5.10 shows the comparison of the results for these networks.

| MAV | RMS | WL | Accuracy [%] |
|---|---|---|---|
| 1 | 0 | 0 | 94.70 |
| 1 | 1 | 0 | 94.70 |
| 1 | 1 | 1 | 96.60 |

**Table 5.10**: Raw results of the networks with different features functions, namely $rN_{10}$, $rN_{14}$, and $rN_5$.

## 5.4.2. Analysis with Condition Parameters

In this thesis, the Condition Parameters is used to give the definition of correct recognized gesture stricter requirements.

### 5.4.2.1. The Condition Parameters

The system have three condition parameters for analysis. The condition parameters are defined as Gesture Margin ($gm$), Difference Margin ($dm$), and Value Threshold ($vt$). The output values given in table 5.11 are used as an example to explain the condition parameters.

| File: recorded-THANKYOU-520.json | |
|---|---|
| EAT | 0.008738 |
| HELP | 0.000742 |
| SLEEP | 0.125932 |
| THANKYOU | 0.364630 |
| WHY | 0.000061 |
| NO | 0.000079 |
| YES | 0.000001 |
| DRINK | 0.001851 |
| HELLO | 0.695881 |
| SORRY | 0.000000 |

**Table 5.11**: The output values of a forward propagation of network $rN_4$. The file consist of data from a *THANKYOU* gesture.

Consider $gm = a$, then the analysis considers it a success if the right gesture is among the gestures with the $a$ highest output value. If $gm = 2$, then the the analysis will consider the given example a success, since $o_{THANKYOU}$ is the second highest output value.

The Difference Margin, say how far the output value of the recognized gesture have to be, to be considered. Let $dm = b$ where $0 < b < 1$, then the output value of the recognized gesture $o_0$ have to be $b$ greater than the second highest output value $o_1$. mathematically, this is written as $o_0 \geq o_1 + b$. If $gm > 1$, then $o_0$ is the lowest output value in the group, and $o_1$ is the highest output value of the gestures not in the group. If $o_0 < o_1 + b$, then the system takes out the gesture with the lowest output, and sets the $o_0 \rightarrow o_1$. The new $o_0$ is the lowest value in the new group. For instance, let $dm = 0.3$ and $gm = 2$, then the analysis will not consider it a success, since $o_{THANKYOU} < o_{SLEEP} + 0.3$, but analysis conider that the network recognized the data as a *HELLO* gesture since $o_{HELLO} \geq o_{THANKYOU} + 0.3$.

The Value Threshold is the threshold for the output value of gesture $g$ to considered $g$ in the group of potential gestures. Let $vt = c$, then $g$ is only considered as a potential gesture if $o_g \geq c$. For example, if $vt = 0.5$ then *THANKYOU* would not be a potential gesture, while *HELLO* would be a potential gesture.

For the values $gm = 1, dm = 0.7, vt = 0.9$, this thesis uses the term strictest condition parameter setting.

#### 5.4.2.2. Results with Condition Parameters

Unlike section 5.4.1, this section dose not presents the results in categories, but shows a list of tables with the results from different networks with different values for the condition parameters.

Tables 5.12 to 5.16 show results from some of the hacakthon dataset based networks, while tables 5.17 to 5.20 shows the results for some of the recorded dataset based networks.

Although the results are not given in any categorization, there are still some underlaying categories. The difference of the networks with the results given in tables 5.12 to 5.15, are the number of hidden layers. Tables 5.16 and 5.18 show the results for networks using only the raw EMG signals. Tables 5.17, 5.20 and 5.21 shows results from networks using different features (MAV, RMS, WL). The difference of the networks with the results presented in tables 5.17 and 5.19 is the wavelet level.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 78.53 |
| 2 | 0.00 | 0.00 | 89.93 |
| 1 | 0.00 | 0.70 | 66.60 |
| 1 | 0.00 | 0.90 | 54.33 |
| 1 | 0.70 | 0.90 | 50.93 |

**Table 5.12**: The result for different values of the parameters, using a network with 1 hidden layers and wavelet level 1, $hN_9$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 76.50 |
| 2 | 0.00 | 0.00 | 87.83 |
| 1 | 0.00 | 0.70 | 69.77 |
| 1 | 0.00 | 0.90 | 64.70 |
| 1 | 0.70 | 0.90 | 59.90 |

**Table 5.13**: The result for different values of the parameters, using a network with 2 hidden layers and wavelet level 1, $hN_8$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 76.43 |
| 2 | 0.00 | 0.00 | 87.70 |
| 1 | 0.00 | 0.70 | 70.00 |
| 1 | 0.00 | 0.90 | 65.03 |
| 1 | 0.70 | 0.90 | 61.60 |

**Table 5.14**: The result for different values of the parameters, using a network with 3 hidden layers and wavelet level 1, $hN_5$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 76.97 |
| 2 | 0.00 | 0.00 | 88.27 |
| 1 | 0.00 | 0.70 | 72.07 |
| 1 | 0.00 | 0.90 | 68.53 |
| 1 | 0.70 | 0.90 | 65.80 |

**Table 5.15**: The result for different values of the parameters, using a network with 4 hidden layers and wavelet level 1, $hN_7$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 78.70 |
| 2 | 0.00 | 0.00 | 90.43 |
| 1 | 0.00 | 0.70 | 63.60 |
| 1 | 0.00 | 0.90 | 46.20 |
| 1 | 0.70 | 0.90 | 45.10 |

**Table 5.16**: The result for different values of the parameters, using a network with 2 hidden layers and the raw EMG signals, $hN_{11}$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 96.60 |
| 2 | 0.00 | 0.00 | 98.00 |
| 1 | 0.00 | 0.70 | 92.30 |
| 1 | 0.50 | 0.70 | 91.30 |
| 1 | 0.00 | 0.90 | 85.50 |
| 1 | 0.70 | 0.90 | 84.50 |

**Table 5.17**: The result for different values of the parameters, using a network with 2 hidden layers and wavelet level 3, $rN_5$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 90.80 |
| 2 | 0.00 | 0.00 | 93.80 |
| 1 | 0.00 | 0.70 | 84.10 |
| 1 | 0.50 | 0.70 | 82.40 |
| 1 | 0.00 | 0.90 | 71.90 |
| 1 | 0.70 | 0.90 | 70.90 |

**Table 5.18**: The result for different values of the parameters, using a network with 2 hidden layers and only the raw EMG signals, $rN_4$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 94.80 |
| 2 | 0.00 | 0.00 | 96.30 |
| 1 | 0.00 | 0.90 | 82.60 |
| 1 | 0.70 | 0.90 | 81.80 |

**Table 5.19**: The result for different values of the parameters, using a network with 2 hidden layers and wavelet level 1, $rN_3$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 94.70 |
| 1 | 0.00 | 0.70 | 88.00 |
| 1 | 0.00 | 0.90 | 71.50 |
| 1 | 0.70 | 0.90 | 69.90 |

**Table 5.20**: The result for different values of the parameters, using a network with 2 hidden layers, wavelet level 3, and only the MAV feature $rN_{10}$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 94.70 |
| 1 | 0.00 | 0.70 | 89.70 |
| 1 | 0.00 | 0.90 | 82.40 |
| 1 | 0.70 | 0.90 | 80.10 |

**Table 5.21**: The result for different values of the parameters, using a network with 2 hidden layers, wavelet level 3, and MAV and RMS features, $rN_{14}$.

### 5.4.3. Gesture Accuracy

This section will give results from a more micro level, and present the results of the accuracy for each gesture, instead an overall presentation. The strict condition parameters and the networks $rN_5$, $rN_4$, and $rN_3$ will be used. The result from the hackathon dataset will not be presented, since not enough information is given about the data. The results are shown in table 5.22.

| Gestur ID | Gesture Name | $rN_5$ [%] | $rN_4$ [%] | $rN_3$ [%] |
|---|---|---|---|---|
| 0 | EAT | 79.00 | 69.00 | 75.00 |
| 1 | HELP | 75.00 | 64.00 | 79.00 |
| 2 | SLEEP | 86.00 | 82.00 | 87.00 |
| 3 | THANKYOU | 90.00 | 76.00 | 92.00 |
| 4 | WHY | 87.00 | 76.00 | 89.00 |
| 5 | NO | 82.00 | 66.00 | 90.00 |
| 6 | YES | 89.00 | 86.00 | 90.00 |
| 7 | DRINK | 79.00 | 71.00 | 81.00 |
| 8 | HELLO | 93.00 | 67.00 | 78.00 |
| 9 | SORRY | 75.00 | 52.00 | 68.00 |

**Table 5.22**: The results for different gestures for different networks, using the strict condition parameter settings $gm = 1$, $dm = 0.7$, $vt = 0.9$

### 5.4.4. Training Set Size

The results of a network trained with a few instances is given in table 5.23.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 60.10 |
| 1 | 0.70 | 0.90 | 42.80 |

**Table 5.23**: The results for different values of the parameters, using a network with 2 hidden layers and wavelet level 3, $rN_0$.

### 5.4.5. Global Normalization

As mentioned in section 4.4.3, the normalization of groups of the input seems to generate a bug. The bug creates duplicated values in the input. This section present the results of a network using a global normalization method. The normalization function is the same as the one explained in section 4.4.3, but $a$ is an array with all the calculated values instead of a fragment of input. Table 5.24 shows the results with different condition parameter settings.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 93.40 |
| 1 | 0.00 | 0.70 | 86.60 |
| 1 | 0.00 | 0.90 | 70.90 |
| 1 | 0.70 | 0.90 | 69.30 |

**Table 5.24**: The results for different values of the parameters, using a network with 2 hidden layers, wavelet level 3, and a global input normalization, $rN_{13}$.

### 5.4.6. "NotMe" Dataset

The "NotMe" dataset, is data sampled from different users than the user of standard recorded dataset. The dataset is sampled from 7 different users. Table 5.25 shows the results from the "NotMe" dataset on the current most accurate network, namely $rN_5$. Table 5.26 shows the accuracy for each gesture on $rN_5$, using $vt = 0.7$.

| Gesture margin | Difference margin | Value threshold | Accuracy [%] |
|---|---|---|---|
| 1 | 0.00 | 0.00 | 36.83 |
| 2 | 0.00 | 0.00 | 57.78 |
| 1 | 0.00 | 0.70 | 21.90 |
| 1 | 0.50 | 0.70 | 20.00 |
| 1 | 0.00 | 0.90 | 13.65 |
| 1 | 0.70 | 0.90 | 12.38 |

**Table 5.25**: The result for "NotMe" dataset with different values of the parameters, using a network with 2 hidden layers and wavelet level 3, $rN_5$.

| Gestur ID | Gesture Name | Accuracy [%] |
|---|---|---|
| 0 | EAT | 37.14 |
| 1 | HELP | 21.42 |
| 2 | SLEEP | 9.38 |
| 3 | THANKYOU | 28.00 |
| 4 | WHY | 13.79 |
| 5 | NO | 37.93 |
| 6 | YES | 37.14 |
| 7 | DRINK | 2.67 |
| 8 | HELLO | 18.42 |
| 9 | SORRY | 15.38 |

**Table 5.26**: The result for ”NotMe” dataset for different gestures, using a network with 2 hidden layers and wavelet level 3, and condition parameter $vt = 0.7$, $rN_5$.

# Chapter 6

# Discussion

This chapter provides a more detailed discussion of the results presented in chapter 5 and a review of the general system.

## 6.1. Result Analysis

This section aims to give a more detailed analysis of the results presented in chapter 5. The performance is discussed in section 6.1.1 and the accuracy in section 6.1.2

### 6.1.1. Performance

While the run time of the forward propagation was relatively constant (dependent on the hardware), the training time varied with the change of network structure. Therefore, it is more interesting to analyse the training time regarding the network structure. However, it is worth mentioning that the reason for relatively constant forward propagation run time can be that the computation time of the input values dominates the run time, and one iteration of the forward propagation is not significant. It is hard to give a precise performance analysis, since the performance is affected by the background tasks.

#### 6.1.1.1. CPU vs GPU

Before the actual discussion of the results, it is worth looking at the performance improvement from the GPU version of TensorFlow.

TensorFlow supports a GPU implementation. It is worth mentioning, that the GPU implementation of TensorFlow utilize both computing power from the CPU and GPU. The

comparison of the CPU and GPU implementation, is done only on the desktop. The laptop has a considerable weaker CPU than the desktop.

Testing for both the hackathon dataset and the recorded dataset, the framework seems to achieve a performance boost of $200\,\%$ to $300\,\%$, depending on the network structure. For instance, the training time using the hackathon dataset with $5000$ epochs on a network with the structure given by

Layer Sizes [120, 150, 150, 6]

was approximately $3\,\mathrm{minutes}$ for the GPU version, and $7\,\mathrm{minutes}$ for the CPU version. While the training with the same criteria, for the network with the structure

Layer Sizes [120, 300, 300, 6]

was approximately $3.5\,\mathrm{minutes}$ for the GPU version, and $13\,\mathrm{minutes}$ for the CPU version. The performance boost of the training seems to be dependent on the size of the network.

With a relativaly constant about $0.1\,\mathrm{seconds}$, the forward propagation did not seem to get any noticeable performance improvement. The forward propagation performance includes the computing of the input values, and this computation is not implemented for the GPU, therefor the whole progress of categorizing a gesture data may be dominated by the computation of the input values and not the forward propagation itself.

#### 6.1.1.2. Depth

Given by table 5.2, there is no significant changes in the run times. The increase of run time with the higher values of $n$ could be random, since the run time are computed from an average of a small quantity, and the run time for running $5000$ epochs were not always higher for network with $n_i$, compared to a network with $n_k$, where $n_i > n_k$. However, it may have a greater affect on the run time if $h_i \gg 150$, since $150$ is a relatively small value. For $h_i = 150$ it seems like the performance is more affected by the external factors.

#### 6.1.1.3. Number of Inputs

Varying the number of input had a much greater impact on the training time. Given in table 5.3, the training time a relatively clear increase with the number of inputs. An increase of the input size $I$ from $24$ to $216$, leads to an increase of training time with over $3$ times. Considering that the training file is pre-generated and all the inputs are pre-computed, we can ignore the input computing time.

#### 6.1.1.4. Size of the Hidden Layers

The training time of networks with different size of the hidden layers is shown in table 5.4. While there is not an equally clear increase of run time, as with the input size given in

table 5.3, there is still a notable increase. The run time of $h = 150$ is higher than $h = 200$ may be due to the external factors, since an increase of $50$ neurons of each layer is relativaly small.

## 6.1.2. Accuracy

The accuracy is the most important value, and is one of the factor that determines if the framework has a potential for a real world usability. With a comprehensive overview, the network based on the hackathon dataset have an accuracy over $75\,\%$, while the network based on the recorded dataset have an accuracy over $90\,\%$, that is for the raw analysis. This section discusses the accuracy of the networks divided into categories determined by parameters such as network depth, wavelet level and features. The results with the condition parameters is used to measure the certainty of the raw results. Note that not all the results used in this section is represented in section 5.4.2.2.

### 6.1.2.1. Depth

By the raw analysis, the network depth seems have little impact on the accuracy. Given by tables 5.6 and 5.7, there are no significant difference in accuracy. However, analysis with the strictest condition parameter settings, given in tables 6.1 and 6.2, shows a greater difference. Tables 5.12 to 5.15 shows that noticeable difference dose not occur before $dm = 0.7$. Meaning the error rate of the networks are lower for the deeper networks.

| $n$ | Accuracy [%] |
|---|---|
| 1 | 50.93 |
| 2 | 59.90 |
| 3 | 61.60 |
| 4 | 65.97 |

**Table 6.1**: Results from the same networks as in table 5.6, namely $hN_9$, $hN_8$, $hN_5$, and $hN_7$, but with strict condition parameters $gm = 1, dm = 0.7, vt = 0.9$. $n$ is the number of hidden layers.

### 6.1.2.2. Wavelet Level

As with the different network depth, there no significant difference of the values in tables 5.8 and 5.9, except for the $rN_4$ in table 5.9, which is a network that only use the raw EMG signals. That the network $hN_{11}$, also only using the raw EMG signals, have a similar result as the other results in table 5.8 may be due to a coincidences.

| $n$ | Accuracy [%] |
|---|---|
| 1 | 71.90 |
| 2 | 84.50 |

**Table 6.2**: Results from the same networks as in table 5.7, namely $rN_7$, and $rN_5$, but with strict condition parameters $gm = 1, dm = 0.7, vt = 0.9$. $n$ is the number of hidden layers.

The tables 6.3 and 6.4 is the results of the same network as in tables 5.8 and 5.9, but with the strict condition parameter settings. From these values, it is hard to tell any pattern for change of the wavelet level, except for wavelet level 0. Wavelet level 0 means excluding the wavelet transform, and use the raw EMG signals. We can see from the tables 6.3 and 6.4 that we obtain a noticeable improved results given the wavelet level $i$ is higher than 0, that is $i > 0$. Even though in table 5.9, $i = 0$ get a better result, it seems like the wavelet transform creates a more certain result.

| wavelet level | Accuracy [%] |
|---|---|
| 0 | 48.10 |
| 1 | 59.90 |
| 3 | 58.63 |

**Table 6.3**: Results from the same networks as in table 5.8, namely $hN_{11}$, $hN_8$, and $rN_6$, but with strict condition parameters $gm = 1, dm = 0.7, vt = 0.9$.

| wavelet level | Accuracy [%] |
|---|---|
| 0 | 70.90 |
| 1 | 81.80 |
| 2 | 80.70 |
| 3 | 84.50 |

**Table 6.4**: Results from the same networks as in table 5.9, namely $rN_4$, $rN_3$, $rN_6$, and $rN_5$, but with strict condition parameters $gm = 1, dm = 0.7, vt = 0.9$.

#### 6.1.2.3. Features

A comparison of network with different feature functions is given in table 5.10. Also here it is difficult to say any pattern with certainty. Note that the feature functions are not tested

separately, rather the feature functions is added for $rN_{10}$, $rN_{14}$, and $rN_5$, respectively. Given the values from tables 5.17, 5.20 and 5.21, a comparison table with the strict condition parameter settings is given in table 6.5. Given by table 5.20, the large drop of accuracy occur when the value threshold increases from $vt = 0.7$ to $vt = 0.9$. The accuracy with $vt = 0.7$ for the three networks are relatively close.

| MAV | RMS | WL | Accuracy [%] |
|---|---|---|---|
| 1 | 0 | 0 | 69.90 |
| 1 | 1 | 0 | 80.70 |
| 1 | 1 | 1 | 85.50 |

**Table 6.5**: Results from the same networks as in table 5.10, namely $rN_{10}$, $rN_{14}$, and $rN_5$, but with strict condition parameters $gm = 1$, $dm = 0.7$, $vt = 0.9$.

#### 6.1.2.4. Training Set Size

Table 5.23 shows the result for network $rN_0$, that only uses a small training set of less than 150 instance per gesture. The accuracy is not too low, but the accuracy of the networks with the bigger training set are twice as high. This thesis does not provide any thoroughly analysis of the number of training instance a network need before it get reliable.

#### 6.1.2.5. Gesture Accuracy

Gesture Accuracy, means the accuracy of the gestures separately. Table 5.22 present the accuracy for all the gestures for the networks $rN_5$, $rN_4$, and $rN_3$. The $rN_4$ is a network using only the raw EMG signals. From the values, it seems like the WT improve the accuracy of some of the gestures, such as the *THANKYOU*, *HELLO* and *NO* gesture.

#### 6.1.2.6. Overall Accuracy

As an overall accuracy with the strict condition parameter settings, the networks based on the hackathon dataset have an accuracy mostly over $50\,\%$, while the networks based on the recorded dataset have an accuracy mostly over $70\,\%$.

$65\,\%$ and $85\,\%$ from the networks based on hackathon dataset and recorded dataset, respectively, are the highest accuracy achieved in this thesis. The networks are $hN_7$ and $rN_5$. The key structure of these networks is the network depth in $hN_7$, with 4 hidden layers, and the wavelet level in $rN_5$. Note that none of the networks based on the recorded dataset is deeper than 2 hidden layers, meaning, the $rN_5$ is the deepest network in it's group. Also the only network with wavelet level 3 in the network based on the hackathon

dataset is $rh_6$. However, this network gives a weaker accuracy than $hN_8$, which has the almost same structure, but with wavelet level 1. This is given in table 6.3.

As mentioned, there was a bug in the grouped normalization, which may affect the results. As an attempt to fix this bug, a global normalization method is used. The method is explained in section 5.4.5, and the results of the network using the global normalization $rN_{13}$ is given in table 5.24. The structure is the same as network $rN_5$, and the results from table 5.17 can be used as a comparison. While $rN_{13}$ have a higher epoch count than $rN_5$, it can be mentioned that the $rN_{13}$ returned approximately the same results at $2500000$ epochs, which is nearly the same as the epoch count for $rN_5$. As $rN_{13}$ produced a much weaker result than $rN_5$ for all tested condition parameter settings, the used global normalization method do not seem to be the fix for the bug.

#### 6.1.2.7. Various Users Accuracy

An user independent system is not the main focus this thesis, and therefore a detailed analysis of is not provided. Table 5.25 present the accuracy table from other users using the network with the highest accuracy, namely $rN_5$. The results are relativly weak, but not totally random. With $gm = 2$, the network achieve an accuracy of over $50\,\%$. Table 5.26 shows the accuracy for different gestures, with $vt = 0.7$, and it seems like the network is able to recognize some of the gestures better than others, especially the *DRINK* gesture has a low accuracy. The network did not give a reliable accuracy for the generalized case. It is not strange that the networks are weak on these data, because the network is trained on a singal user. How the gesture should be done varies from people to people, such as thumb the placement. People also have different size of the forearm, causing the EMG sensors to measure EMG data from different areas of the forearm. The "NotMe" dataset is also relatively small and it is hard to give any reliable conclusion to the data.

#### 6.1.2.8. Different Activation Functions

Some of the network provided in appendix E use the ReLu and SoftMax activation function, but these networks produced strange results where some of the output values always returned $0$, thus this thesis choose to focuses on the sigmoid activation function only.

## 6.2. System Analysis

This section provides an analysis of the limitation of the framework given in section 6.2.1, and section 6.2.2 gives a comparison with the results of the previous work.

### 6.2.1. Limitations

The framework for this thesis is a prototype level system, and includes strict constraints and limitation. The framework is a isolated gesture recognition system, which means it cannot separate a sequence of gestures. The framework isolates the gestures with a time interval, starting from when the user press the initialization button. This time interval is constant, and the framework have no method to locate the relevant gesture data.

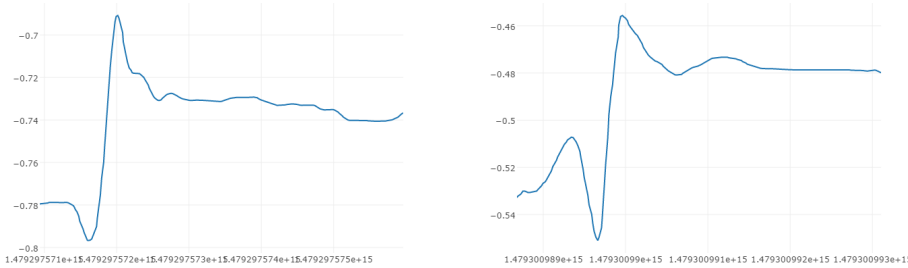
The framework requires that the Myo armband should be placed on an approximately fixed location of the forearm. This causes the user to be cautious of the placement of the Myo armband. The framework is not able to detect which arm the Myo armband is on, but this functionality is already pre-implemented in the myo SDK, making this functionality simple to integrate.

### 6.2.2. Comparison with the Previous Work

The preliminary framework, described in section 4.1, used cross correlation and DTW to find similarity to known gesture data. Note that the test for the previous work was relatively small, and the results may not be too reliable. The results for cross correlation and DTW had a relatively good accuracy on the IMU data, but poor accuracy for the EMG data. It does not seem like method to measure the similarity of two series work with EMG signals. In figure 6.1, the visual similarity of figures 6.1a and 6.1b is shown. These graphs represent one of the axis of the orientation given by the magnetometer. Cross correlation and DTW seem to work well for these graphs. As seen in figures 6.1a and 6.1b, there exist a visual similarity, but the signals are noisy. The noise seems to affect the result of cross correlation and DTW, and these methods dose not preform well on the EMG signals. From the results of this thesis, feature extraction and machine learning classification seem to be a relatively good solution for classification of EMG signals, compared to cross correlation and DTW. The current framework, that only utilize the data from the EMG sensors, was able to get an accuracy higher than the accuracy of the preliminary framework. Note that the results of the current system is more reliable with a greater quantity. It is worth to mention that the current system is able to recognize $10$ different gestures, while the previous system was only able to recognize $5$.

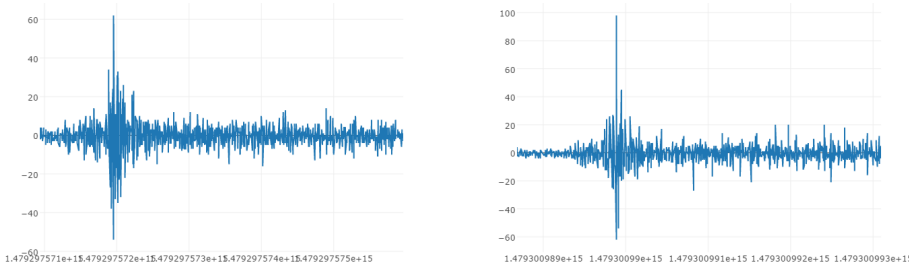
Performance-wise, the preliminary framework used over $10\,\mathrm{seconds}$ to analyze a gesture, while the current framework use under $0.2\,\mathrm{seconds}$. Note this performance is given from the same hardware, the laptop. The preliminary framework was implemented using a non-optimized algorithm for cross correlation and DTW, but an improvement of over $5000\,\%$ would be hard to beat even with an optimized algorithm.

In both performance and accuracy, the current framework gave better results. However, the preliminary framework has an advantage that it do not require a large quantity of training instances. The preliminary framework used $3$ known gesture data to measure the similarity of the unknown data. The current framework, on the other hand, requires a




(a) (b)

**Figure 6.1**: The graphs given in (a) and (b) are two independent graphs, representing data from one of the axis of the orientation data given by the magnetometer.




(a) (b)

**Figure 6.2**: The graphs given in (a) and (b) are two independent graphs, representing the EMG data from the same sensor of the same gesture.

sufficient amount of training, and for the produced results, this amount was a training set of 500 instance per gesture.

# Chapter 7

# Conclusion and Future Work

This chapter provides a summary of this thesis, and includes a review of the framework and the main reasons for the choices. Future improvments of this framework are given in section 7.2.

## 7.1. Conclusion

An ideal EMG based gesture recognition system would be an EMG signal analysis system which is accurate, simple, fast and reliable. The limitations presented in section 6.2.1 makes the framework presented in this thesis unpractical, not for practical uses that is. However, the scope of this thesis is not to develop a system for a practical purpose, but to explore the features of EMG signals, and utilize different structures of neural networks to classify gesture data based on these features.

This thesis propose the development of a framework for gesture recognition which utilize surface EMG. The framework uses the Myo armband to measure the EMG signals from the arm. While the Myo armband also provides IMU data, only utilization of the EMG data is implemented in the framework. The previous work, which the framework is based on, proposed relatively good results for gesture recognition using IMU data by using methods to measure similarity between graphs. However, unlike IMU data, the EMG signals contains a lot of noises. For EMG signals, which are complex sets of data, feature extraction is presumably a better solution.

The purpose of this thesis is to look at the potentials of using deep learning to classify EMG data. The framework only supports the simplest neural network architecture, the fully connected feed forward neural networks. This thesis experiments with different types of network structures, with the aim to find out which parameters are important for the accuracy, and possibly preformance.

Raw EMG signals are represented by the time-amplitude representation, but distinguished information can be hidden in the frequency content of the signal. This thesis uses the wavelet analysis to extract frequency dependent features from the EMG signals. The Wavelet Transform captures both frequency and location information, which is an advantage when working with non-stationary signals such as the EMG signals. The results shows that the wavelet analysis improve the accuracy.

The achieved accuracy was relatively high, but the networks trained in this thesis is user dependent, and dose not achieve a high accuracy for the generalized case. All the network achieved an accuracy over $90\,\%$ with the least strict classification method, while most of the networks structures achieved an accuracy over $75\,\%$ with stricter conditions. The highest accuracy with the strict condition was $85\,\%$. The framework shows good performance with an analyze time below $0.2\,\mathrm{seconds}$ for each gesture data. This gives the framework, performance-wise, a good base for a practical purpose.

The results of this thesis is relatively good, and even though the data was sampled from a single individual, the application still have great potentials. This thesis also deals with a larger and maybe more generalized dataset, but since there is limited information about this dataset, the results comes with a little uncertainty. The framework is still a prototype-level system, and is far from practical system. Possible future improvements of the framework is discussed in section 7.2.

## 7.2. Future Work

This section provides possible improvement and potentials to the framework.

### 7.2.1. Normalization Bug

As explained in section 4.4.3, there is a bug with the normalization. While the framework produced relatively good results with the bug, some of the data become redundant since the bug create duplication of the values. It is not clear why this bug occur, but it looks like it occur from the grouped normalization. A global normalization was used as a quick fix for the bug, but as shown in section 5.4.5, the fix did not improve the results.

### 7.2.2. Two-Handed Gestures

The framework only support one Myo device, but as mentioned in section 2.1, ASL utilize both hands. An improvement for the framework would be to implement support for analyzing data from two Myo armbands. This may lead to an improvement in accuracy, because the framework receive more information about the gestures. Note that while most of the gesture in this thesis are one-handed gestures, the gesture *HELP* use the left hand as a supporting factor.

### 7.2.3. IMU Utilization

The Myo armband provides data from both IMU and EMG sensors, but this thesis only utiliz the data provided by the EMG sensors. With information of the position and orientation of the arm, it will be easier to classify the gesture. The framework, from the previous work described in section 4.1, utilized the IMU data to classify gestures by using cross correlation and DTW. While the analysis from the previous work produced relatively good results for the IMU data, the performance was relatively slow compared to the current framework.

The current framework is dependent on that the EMG sensor have to sample EMG data from approximately the same position on the forearm. By using IMU data, it is possible to determined the position of the Myo armband on the forearm, and remove one constrain from the user. The SCEPTERE prosject [44], includes a method to determind the position of the forearm.

### 7.2.4. User Independence

The discussion of the results from the ”NotMe” dataset is presented in section 6.1.2.7. The networks trained for this thesis is based on a dataset sampled from one single user, and the accuracy of the ”NotMe” dataset was relatively low compared to the other files in the recorded dataset. While this thesis also present results given by the networks trained with the hackathon dataset, it is not reliable since not enough information are given about the quality of the data. To improve the reliability of the results, a more generalized dataset have to be used.

### 7.2.5. Continuous Gesture Recognition

Gesture Recognition can be categorized as Continuous or Isolated Gesture Recognition. The framework is an isolated gesture recognition system, and does not provide an on-demand recognition functionally. An initial button is used to tell the framework to start the sampling of data. This method helps the framework to isolate the gesture data, and there is no need for the framework to locate the relevant data. Segmentation of a word is important in gesture recognition because there may be many meaningless movements between words. For the framework to have a practical propose, it must be able to distinguish between gesture and non-gesture movements, and segment a sequence of words.

Researchs on continuous gesture recognition based on the Hidden Markov Model (HMM) are presented in [33, 16, 10]. In [38], a system utilizing a start posture and a Recurrent Neural Network (RNN) architecture to achieve an continuous gesture recognition is presented. Note that these researches are not based on EMG data, but orientations [33, 38] and spatio-temporal trajectories [16, 10].

Continuous speech recognition is a close related field, and is more developed. There are a lot of related researches on speech recognition based on ANN. In [51, 24], a research on continuous speech recognition using RNN are presented. A hybrid HMM and HMM approach is presented in [37].

### 7.2.6. Network Architectures

This thesis is based on the simplest deep network architecture. As shown in figure 2.8 there are many different network architectures, which have different advantages and disadvantages. The RNNs, described in section 2.4.5.3, has the ability to remember and make use of the sequential information. This feature is useful for analyzing a sequence of gestures. And as mentioned in section 7.2.5, RNNs has its applications within the continuous gesture recognition.

The Convolutional Neural Network (CNN) architecture, described in section 2.4.5.2, is an architecture widely used in image analysis. This networks are distinguishable by the ability to extract features of local segments of the data, instead of using a global feature extraction. While CNNs are widely used for image analysis, such as hand writing recognition, it also has its applications on time-series data, such as speech recognition and gesture recognition. A CNN based speech recognition is presented in [1, 50] and a CNN based classification method of movements for prosthetic hands using EMG data is presented in [4].

# References


[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] Antonios K Alexandridis and Achilleas D Zapranis. Wavelet neural networks: A practical guide. *Neural Networks*, 42:1–27, 2013.

[3] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. EMG-based control of a robot arm using low-dimensional embeddings. *IEEE Transactions on Robotics*, 26(2):393–398, 2010.

[4] Manfredo Atzori, Matteo Cognolato, and Henning Müller. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in Neurorobotics*, 10, 2016.

[5] Cynthia J Kellett Bidoli and Elana Ochse. *English in international deaf communication*, volume 72. Peter Lang, 2008.

[6] Blargg. The Nyquist Limit. http://www.slack.net/~ant/bl-synth/3.nyquist.html. Accessed: 28.05.2016.

[7] Baptiste Caramiaux, Marco Donnarumma, and Atau Tanaka. Understanding gesture expressivity through muscle sensing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(6):31, 2015.

[8] Xiujuan Chai, Guang Li, Yushun Lin, Zhihao Xu, Yili Tang, Xilin Chen, and Ming Zhou. Sign language recognition and translation with kinect. In *IEEE Conf. on AFGR*, 2013.

[9] Adrian Dart Cheong Chan, Kevin Englehart, Bernard Hudgins, and Dennis Fenton Lovely. Hidden markov model classification of myoelectric signals in speech. *IEEE Engineering in Medicine and Biology Magazine*, 21(5):143–146, 2002.

[10] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 21(8):745–758, 2003.

[11] Ching-Hua Chuan, Eric Regina, and Caroline Guardino. American sign language recognition using leap motion sensor. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 541–544. IEEE, 2014.

[12] Liu Chun-Lin. A tutorial of the wavelet transform. *NTUEE, Taiwan*, 2010.

[13] CNET. Myo armbands used to control prosthetic arm. https://www.cnet.com/news/myo-armbands-used-to-control-prosthetic-arm/. Accessed: 02.12.2016.

[14] Ayushman Dash. Pewter. https://github.com/sigvoiced/pewter. Accessed: 05.12.2016.

[15] Tim Dettmers. Deep Learning in a Nutshell: Core Concepts. https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/. Accessed: 08.06.2017.

[16] Mahmoud Elmezain, Ayoub Al-Hamadi, Jorg Appenrodt, and Bernd Michaelis. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[17] Ian Esper. Convolutional Neural Network for Hand Gesture Recognition Using Myo. https://github.com/imesper/CNNHandGesture/tree/master/src.

[18] Dario Farina, Roberto Merletti, and Roger M Enoka. The extraction of neural strategies from the surface EMG. *Journal of Applied Physiology*, 96(4):1486–1495, 2004.

[19] Osamu Fukuda, Toshio Tsuji, Akira Ohtsuka, and Makoto Kaneko. EMG-based human-robot interface for rehabilitation aid. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3492–3497. IEEE, 1998.

[20] Maestro Gesture Glove. http://maestroglove.com/. Accessed: 03.12.2016.

[21] Camron Godbout. Recurrent Neural Networks for Beginners. https://medium.com/@camrongodbout/recurrent-neural-networks-for-beginners-7aca4e933b82. Accessed: 26.05.2017.

[22] Susan Goldin-Meadow. The role of gesture in communication and thinking. *Trends in cognitive sciences*, 3(11):419–429, 1999.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[24] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[25] Akira Hiraiwa, Katsunori Shimohara, and Yukio Tokunaga. Emg pattern analysis and classification by neural network. In *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 1113–1115. IEEE, 1989.

[26] Xuedong D Huang, Yasuo Ariki, and Mervyn A Jack. *Hidden Markov models for speech recognition*, volume 2004. Edinburgh university press Edinburgh, 1990.

[27] J.R. Johansson. Introduction to scientific computing with Python. http://nbviewer.jupyter.org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-0-Scientific-Computing-with-Python.ipynb. Accessed: 27.05.2016.

[28] Jonghwa Kim, Stephan Mastnik, and Elisabeth André. EMG-based hand gesture recognition for realtime biosignal interfacing. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 30–39. ACM, 2008.

[29] Kinect for Xbox. http://www.xbox.com/en-US/xbox-one/accessories/kinect. Accessed: 05.12.2016.

[30] Simon Lang, Marco Block, and Raúl Rojas. Sign language recognition using kinect. In *International Conference on Artificial Intelligence and Soft Computing*, pages 394–402. Springer, 2012.

[31] Leap Motion. https://www.leapmotion.com/. Accessed: 05.12.2016.

[32] Ki-Seung Lee. Emg-based speech recognition using hidden markov models with global control variables. *IEEE Transactions on Biomedical Engineering*, 55(3):930–940, 2008.

[33] Rung-Huei Liang and Ming Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 558–567. IEEE, 1998.

[34] Amol Lolure and VR Thool. Wavelet transform based emg feature extraction and evaluation using scatter graphs. In *Industrial Instrumentation and Control (ICIC), 2015 International Conference on*, pages 1273–1277. IEEE, 2015.

[35] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1565–1569. IEEE, 2014.

[36] Hrushikesh Narhar Mhaskar and Charles A Micchelli. How to choose an activation function. *Advances in Neural Information Processing Systems*, pages 319–319, 1994.

[37] Nelson Morgan and Herve Bourlard. Continuous speech recognition. *IEEE signal processing magazine*, 12(3):24–42, 1995.

[38] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242. ACM, 1991.

[39] Wai Keng Ngui, M Salman Leong, Lim Meng Hee, and Ahmed M Abdelrhman. Wavelet analysis: mother wavelet selection methods. In *Applied mechanics and materials*, volume 393, pages 953–958. Trans Tech Publ, 2013.

[40] Steffen Nissen. FANN: Fast Artificial Neural Network Library. http://leenissen.dk/fann/wp/.

[41] WFD - World Federation of the Deaf. Sign Language. https://wfdeaf.org/human-rights/crpd/sign-language. Accessed: 26.11.2016.

[42] University of Washington. UW undergraduate team wins $10,000 Lemelson-MIT Student Prize for gloves that translate sign language. http://www.washington.edu/news/2016/04/12/uw-undergraduate-team-wins-10000-lemelson-mit-student-prize-for-gloves-that-translate-sign-language/ . Accessed: 03.12.2016.

[43] Christopher Olah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 26.05.2017.

[44] Prajwal Paudyal, Ayan Banerjee, and Sandeep KS Gupta. SCEPTRE: a Pervasive, Non-Invasive, and Programmable Gesture Recognition Technology. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 282–293. ACM, 2016.

[45] J Pauk. 419. different techniques for emg signal processing. *Signal Processing*, 2008.

[46] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. Feature reduction and selection for emg signal classification. *Expert Systems with Applications*, 39(8):7420–7431, 2012.

[47] Robi Polikar. The wavelet tutorial, 1996.

[48] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The leap motion controller: a view on sign language. In *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, pages 175–178. ACM, 2013.

[49] PyWavelet. PyWavelet. https://pywavelets.readthedocs.io.

[50] Yanmin Qian and Philip C Woodland. Very deep convolutional neural networks for robust speech recognition. *arXiv preprint arXiv:1610.00277*, 2016.

[51] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neural networks in continuous speech recognition. In *Automatic speech and speaker recognition*, pages 233–258. Springer, 1996.

[52] Stuart Russell, Peter Norvig, and Artificial Intelligence. Artificial intelligence: A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

[53] Sachin Sharma, Gaurav Kumar, Sandeep Kumar, and Debasis Mohapatra. Techniques for feature extraction from emg signal. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(1), 2012.

[54] Abdulhamit Subasi, Mustafa Yilmaz, and Hasan Riza Ozcalik. Classification of emg signals using wavelet neural network. *Journal of neuroscience methods*, 156(1):360–367, 2006.

[55] TensorFlow. TensorFlow. https://www.tensorflow.org/.

[56] Thalmic. Myo Armband. https://www.myo.com/. Accessed: 17.10.2016.

[57] Thalmic. Myo SDK Documentation. https://developer.thalmic.com/docs/api_reference/platform/index.html. Accessed: 10.11.2016.

[58] Theano. Theano. http://deeplearning.net/software/theano/.

[59] American Sign Language University. American Sign Language Dictionary. http://www.lifeprint.com/.

[60] Clemens Valens. A really friendly guide to wavelets. *ed. Clemens Valens*, 1999.

[61] WebMD. Electromyogram (EMG) and Nerve Conduction Studies. http://www.webmd.com/brain/electromyogram-emg-and-nerve-conduction-studies. Accessed: 24.11.2016.

[62] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton, and Peter Presti. American sign language recognition with the kinect. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 279–286. ACM, 2011.

[63] Xu Zhang, Xiang Chen, Yun Li, Vuokko Lantz, Kongqiao Wang, and Jihai Yang. A framework for hand gesture recognition based on accelerometer and emg sensors. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(6):1064–1076, 2011.

# Figure Source

[64] Concise Visual Summary of Deep Learning Architectures. http://www.datasciencecentral.com/m/blogpost?id=6448529:BlogPost:470286. Retrieved: 31.05.2017.

[65] The Myo Armband. https://www.myo.com/techspecs/. Retrieved: 02.05.2017.

[66] The Myo SDK. https://developer.thalmic.com/docs/api_reference/platform/the-sdk.html. Retrieved: 02.05.2016.

# Appendices

# Appendix A

# Git Repositories

## A.1. Source Code

The git repository for the source code of the current system is

https://github.com/Tonychausan/MyoArmbandPython

## A.2. Dataset

The data set for the hackathon dataset will not be provided, since the dataset was received from the author of [17]. The git repository for the recorded dataset, TensorFlow networks and results analysis is

https://github.com/Tonychausan/MyoArmbandProjectData

# Appendix B

## Gesture Descriptions

A table of all the pre-defined gestures is show in table B.1. While all the gestures are taken from the ASL dictonary [59], it probably contains errors since there is no proper quality check from a qualified sign language user.

| Gestur ID | Gesture Name | Source |
|---|---|---|
| 0 | EAT | http://www.lifeprint.com/asl101/pages-signs/e/eat.htm |
| 1 | HELP | http://www.lifeprint.com/asl101/pages-signs/h/help.htm |
| 2 | SLEEP | http://www.lifeprint.com/asl101/pages-signs/s/sleep.htm |
| 3 | THANKYOU | http://www.lifeprint.com/asl101/pages-signs/t/thankyou.htm |
| 4 | WHY | http://www.lifeprint.com/asl101/pages-signs/w/why.htm |
| 5 | NO | http://www.lifeprint.com/asl101/pages-signs/n/no.htm |
| 6 | YES | http://www.lifeprint.com/asl101/pages-signs/y/yes.htm |
| 7 | DRINK | http://www.lifeprint.com/asl101/pages-signs/d/drink.htm |
| 8 | HELLO | http://www.lifeprint.com/asl101/pages-signs/h/hello.htm |
| 9 | SORRY | http://www.lifeprint.com/asl101/pages-signs/s/sorry.htm |

**Table B.1**: Table of the pre-defined gestures.

# Appendix C

## Computer Specifications

Tables C.1 and C.2 shows the specification of the computers used to train and run the system.

| Laptop | |
|---|---|
| CPU type | Intel Core i7-4500U |
| CPU speed | 1.80 GHz |
| Graphics | Intel HD 4400 |
| OS | Windows 10 Education |

**Table C.1**: Laptop Specifications

| Desktop | |
|---|---|
| CPU type | Intel Core i5-6500 |
| CPU speed | 3.2 GHz |
| Graphics | MSI GeForce GTX 1060, 6 GB GDDR5 |
| OS | Windows 10 Home |

**Table C.2**: Desktop Specifications

# Appendix D

# Data Structure

## D.1. Recorded Dataset Structure

```
{
    "gyr":{
        "data":[
            [ ],
            [ ],
            [ ]
        ]
    },
    "acc":{
        "data":[
            [ ],
            [ ],
            [ ]
        ]
    },
    "emg":{
        "data":[
            [ ],
            [ ],
            [ ],
            [ ],
            [ ],
            [ ],
            [ ],
            [ ]
```

```
            ]
    },
    "ori":{
            "data":[
                [ ],
                [ ],
                [ ],
                [ ]
            ]
     }
}
```

## D.2. Network Input Structure

Consider a function that returns the input values given by some parameters

$$\text{input}(\text{emg\_source},\ \text{level},\ \text{isRecosntructed},\ \text{feature\_id})$$

where the parameter emg_source is the EMG soruce id, the isRecosntructed parameter tells the function if the segment is a coefficient subset or a reconstructed signal. If isRecosntructed is *FALSE*, then the level parameter can be defined as the $i$ in $cD_i$ as given in figure 4.3 and if $i = 0$ then it means the input value is extracted from $cA_i$. If isRecosntructed is *TRUE* than it means it extract the input value from the corresponded reconstructed signal. The feature_id tells the functions to which feature functions to use. if isRecosntructed is $-1$, then the function use the raw EMG signal. Then the input structure of decomposition level $n$ and $m$ number of features is

```
[
    [input(0, 0, TRUE, 0),..., input(8, 0, TRUE, 0)],
    [input(0, 1, TRUE, 0),..., input(8, 1, TRUE, 0)],
    ...
    [input(0, n, TRUE, 0),..., input(8, n, TRUE, 0)],
    [input(0, 0, FALSE, 0),..., input(8, 0, FALSE, 0)],
    [input(0, 1, FALSE, 0),..., input(8, 1, FALSE, 0)],
    ...
    [input(0, n, FALSE, 0),..., input(8, n, FALSE, 0)],
    [input(0, 0, TRUE, k),..., input(8, 0, TRUE, k)],
    [input(0, 1, TRUE, k),..., input(8, 1, TRUE, k)],
    ...
    [input(0, n, TRUE, k),..., input(8, n, TRUE, k)],
    [input(0, 0, FALSE, k),..., input(8, 0, FALSE, k)],
    [input(0, 1, FALSE, k),..., input(8, 1, FALSE, k)],
    ...
    [input(0, n, FALSE, k),..., input(8, n, FALSE, k)],
    [input(0, -1, FALSE, k),..., input(8, -1, FALSE, k)]
]
```

# Appendix E

# Network Structures

This chapter provide the metadata of all the network created for this thesis. The notation is described in section 5.2.4. The system deals with two types of network, one based on the hackathon dataset and the other based on the recorded dataset.

For the network based on recorded datast, some information is missing. The network $rN_{13}$, use a globalized normalization operation as explained in section 5.4.5. And some of the network use a smaller training set than the default described in section 5.2.2, namely $rN_0$, $rN_1$, and $rN_2$. The smaller training set includes the deprecated data files from Pewter, described in section 4.5.2.

## E.1. Hackathon Dataset

| | $hN_0$ |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [120, 150, 24, 6] |
| Epoch count | 2170000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_1$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 2 |
| Layer Sizes | [120, 150, 24, 2] |
| Epoch count | 16950010 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_2$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 2 |
| Layer Sizes | [120, 150, 150, 2] |
| Epoch count | 4205000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_3$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 2 |
| Layer Sizes | [120, 150, 150, 2] |
| Epoch count | 1800000 |
| Activation | ['ReLu', 'ReLu', 'Softmax'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_4$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 2 |
| Layer Sizes | [16, 25, 25, 2] |
| Epoch count | 5280000 |
| Activation | ['ReLu', 'ReLu', 'Softmax'] |
| Wavelet level | 0 |
| [MAV, RMS, WL] | [1, 1, 0] |

| $hN_5$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [120, 150, 150, 150, 6] |
| Epoch count | 8055000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_6$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [216, 300, 300, 6] |
| Epoch count | 10010000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_7$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [120, 150, 150, 150, 150, 6] |
| Epoch count | 10015000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid', 'Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_8$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [120, 150, 150, 6] |
| Epoch count | 10015000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_9$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [120, 150, 6] |
| Epoch count | 10005000 |
| Activation | ['Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $hN_{10}$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 2 |
| Layer Sizes | [16, 30, 30, 2] |
| Epoch count | 5000000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 0 |
| [MAV, RMS, WL] | [1, 1, 0] |

| $hN_{11}$ | |
|---|---|
| Dataset | Hackathon dataset |
| Number of gestures | 6 |
| Layer Sizes | [24, 60, 60, 6] |
| Epoch count | 5000000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 0 |
| [MAV, RMS, WL] | [1, 1, 1] |

## E.2. Recorded Dataset

| $rN_0$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [120, 150, 24, 10] |
| Epoch count | 4970000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_1$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [120, 150, 150, 10] |
| Epoch count | 4030000 |
| Activation | ['ReLu', 'ReLu', 'Softmax'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_2$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [120, 150, 150, 10] |
| Epoch count | 5000000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_3$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [120, 150, 150, 10] |
| Epoch count | 3440000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 1 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_4$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [24, 50, 50, 10] |
| Epoch count | 5000000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 0 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_5$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [216, 300, 300, 10] |
| Epoch count | 2480000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_6$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [168, 200, 200, 10] |
| Epoch count | 2595000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 2 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_7$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [216, 300, 10] |
| Epoch count | 2500000 |
| Activation | ['Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_8$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [144, 150, 150, 10] |
| Epoch count | 0 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 0] |

| $rN_9$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [144, 300, 300, 10] |
| Epoch count | 2500000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 0] |

| $rN_{10}$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [72, 150, 150, 10] |
| Epoch count | 2225000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 0, 0] |

| $rN_{11}$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [8, 10, 10] |
| Epoch count | 0 |
| Activation | ['Sigmoid', 'Sigmoid'] |
| Wavelet level | 0 |
| [MAV, RMS, WL] | [1, 0, 0] |

| $rN_{12}$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [216, 300, 300, 10] |
| Epoch count | 2500000 |
| Activation | ['ReLu', 'ReLu', 'Softmax'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_{13}$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [216, 300, 300, 10] |
| Epoch count | 4000000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 1] |

| $rN_{14}$ | |
|---|---|
| Dataset | Recorded dataset |
| Number of gestures | 10 |
| Layer Sizes | [144, 300, 300, 10] |
| Epoch count | 2500000 |
| Activation | ['Sigmoid', 'Sigmoid', 'Sigmoid'] |
| Wavelet level | 3 |
| [MAV, RMS, WL] | [1, 1, 0] |