UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Recognition of continious arm movement based on electromyography data

Author:
Markiian MATSIUK

Supervisor:
Oleh FARENYUK

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Markiian MATSIUK, declare that this thesis titled, “Recognition of continious arm movement based on electromyography data” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"As computer intelligence gets better, what will be possible when we interface our brains with computers? It might sound scary, but early evidence suggests otherwise: interfacing brains with machines can be helpful in treating traumatic brain injury, repairing spinal cord damage, and countless other applications."

Bill Maris

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Recognition of continious arm movement based on electromyography data

by Markiian MATSIUK

Abstract

Currently, neural-computer interfaces require expensive hardware, which is not available for most researchers, while EMG sensors are cheap, affordable, and quite robust. That makes them an attractive option for a wide class of devices, like prostheses, game devices, or exoskeletons. So reliable and accurate methods of EMG data recognition and interpretation are required. While most of the popular methods of EMG data analysis include only distinct gesture recognition, in this thesis we try to implement the system, which recognizes continuous motion on the example of arm movement and end effector (palm) pose estimation. This thesis goal is to prove that this kind of estimation is possible by creating a system that will estimate arm position in 3d space.

Acknowledgements

I want to thank my supervisor Oleg Farenyuk, who helped with my thesis both in the theoretical and practical part. Also, I want to thank Oles Dobosevych, who helped to organize my working process. Finally, I am grateful to Ukraine Catholic University and especially the Faculty of Applied Science, for its atmosphere and 4-year long journey, for all friends, which I met there, and awesome experience of student life.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	**A**nalog **D**igital **C**onverter
ANN	**A**rtificial **N**eural **N**etwork
DMA	**D**irect **M**emory **A**ccess
DNN	**D**eep **N**eural **N**etwork
DOF	**D**egrees **O**f **F**reedom
EMG	**E**lectro **M**yo **G**raphy
MCU	**Micro**controller **U**nit
Mocap	**Mo**tion **Cap**ture
RGBD	**R**ed **G**reen **B**lue **D**epth
RMS	**R**oot **M**ean **S**quared
RNN	**R**ecurrent **N**eural **N**etwork
ROS	**R**obot **O**perating **S**ystem

Dedicated to all of those, who have a passion for creating something new

Chapter 1

Introduction

1.1 Motivation

While neural-computer interface technology is currently not widely available for many researchers, EMG sensors are cheap and affordable. Therefore they can be used in a variety of solutions, from exoskeletons to game devices. So, precise analysis of data obtained from this type of sensor is necessary in various spheres. Most of the approaches of EMG data analysis are targeted at distinguishing distinct gestures, like in prosthetic arms or experimental game controllers, while this work is describing the approach of analyzing continuous motion.

1.2 Goals

- Verify possibility of recognition of continuous movement from home-made EMG sensors
- Use developed approach to recognize arm movement and produce accurate position model

1.3 Structure

Chapter 2: Related works
This chapter is presenting existing EMG data recognition approaches and algorithms, mainly gesture classifying algorithms.

Chapter 3: Background information
This chapter describes the physics behind EMG, anatomy details, and some signal processing techniques used in this thesis

Chapter 4: System overview
This chapter contains information about system architecture used in experiments, hardware, and software.

Chapter 5: Data overview
This chapter contains analysis and insights on data collected during experiments, rectification, and data preprocessing approaches.

Chapter 6: Methodology
In this chapter proposed approach is described in details, small comparison of different approaches is provided

Chapter 7: Experiment results
This chapter contains an analysis and summary of experiment results.

Chapter 8: Conclusion
This chapter summarizes, discusses further steps and the importance of results obtained in the thesis.

Chapter 2

Related works

The study on EMG analysis started in the early 1950s when the first machine capable of EMG capture was developed by the team of Edward H. Lambert and was small enough to be moved from a laboratory. First machines were not capable of recording EMG data. Basically, it was an oscilloscope connected to sensors, with a Polaroid camera that takes a photo of the screen, but from that point, EMG data analysis becomes possible. Up to the 1980s, all analyses were performed by hand, but since the middle of the 1980s, EMG sensors have become small and cheap enough to become widely used. From the 1980s to 2010s main part of the analysis has been done via classic methods [7], [4]. For example, wavelet analysis is a technique that splits a signal into a combination of basic wavelets, which allows further classifying them [5]. Or analysis in the frequency domain — usage of Fourier transform and phase correlation to find matching EMG patterns [12].

These methods are still used even in recent researches, as they are predictable and robust. But since computational capabilities grow, more and more researchers start using machine learning in EMG analysis. In our work, we also will focus on deep learning approaches.

For the neural network, an important step is dataset generation. As we analyze the continuous signal, it's important to analyze small batches containing ordered data obtained during some time, not the single measurements. There are two different techniques of generating such a dataset — sliding window or disjoint windows. Disjoint windows are a subcase of a sliding window, where an overlapping region is 0, this type of window creates a dataset with data, which is not partially repeating itself, but can lose some information, which can be crucial for the neural network training, so often sliding window approach is used. Size of sliding window is chosen, to provide as small as possible latency, with still high accuracy, typical is 200 window with a shift of 5 up to 100, what gives us delay less than 300, what is acceptable for continuous movement analysis [19].

There are few main techniques used to analyze surface EMG data.

2.1 Artificial neural networks

There are two ways of passing data to neural network - feeding raw signal or using engineered features, which is more robust, providing more precise results. In most of the existing works, engineered features are used in combination with raw signal because it prevents losing any bit of data.

2.1.1 Feature engineering

There are few types of known features that can be engineered for biological signals. For example, Hjorth parameters, which are generally used in EEG signal processing

[16], But can be successfully used for the classification of other biomedical signals, like EMG [20].
Other features are connected with classic signal processing techniques, like discrete wavelet transform or discrete Fourier transform [1].

2.1.2 Neural network

For this type of neural network, the multi-layer perceptron is often used, with different layers configuration [11], which are tuned to work best with provided data.

2.2 Convolution neural network

In this type of network, often 2D data is processed. For example, we can use signal spectrogram, which is signals 2d feature [19], [6].

2.2.1 Feature engineering

Besides the spectrogram, principal component analysis can be used to find the most meaningful features, which will be fed to the neural network to increase stability and speed up the neural network [19]. But often, the raw signal spectrogram is used, as nowadays computers are capable of processing a large amount of 2d data.

2.3 Recurrent neural network

This type of network is known to produce good results, for time-series data, because of its internal feedback loops. So it's obvious to use it for EMG [8]. Moreover, for continuous motion recognition, it's possible to have arm pose (set of joint angles) as an output of the neural network, due to its memory of previous states, instead of velocity (pose change over time) in the case of two previous types of NN.

Chapter 3

Background information

3.1 How EMG sensor works

Electromyography signals — signals obtained from muscle contractions. During contraction, our muscles generate small potential differences, which can be registered using specific sensors. Generally, there are two types of EMG — surface and invasive. In invasive electromyography, small electrodes are inserted into the muscle, which allows us to detect the contraction of single muscle fibers. In contrast, in the case of surface electromyography, electrodes are placed on the skin above the muscle and detect the contraction of a big amount of fibers on some areas[17]. Of course, invasive electromyography gives us more precise information about muscle contraction because we are not dealing with skin resistance, have less noise because the electrode is in direct contact with the muscle fiber. Still, it is significantly harder to perform this type of EMG outside of a lab. Therefore more often, surface EMG is used even if it is not so precise. But sometimes, this issue can become a feature, as we get total muscle contraction information of the muscle instead of single fibers. In the following subsection, we will look at the anatomy and physics of muscle contraction.

3.1.1 Details of anatomy

Every muscle consists of small fibers capable of contraction and relaxation, the combined effort of all fibers creates a force of our muscles. Mainly EMG is used with skeletal muscles (like biceps or triceps). Contraction of this type of muscle is initiated by impulses in the neurons and is usually consciously controlled [9].

Big amount of special type neurons is involved in controlling skeletal muscles. This type of neuron is called "motor neuron," It is usually not in direct contact with muscle tissue but is close to it, stimulating not one but many fibers. This group of neurons and muscle fibers is called a motor unit. Overall our body is electrically neutral, but every neural cell membrane is polarized, so there is a difference in charge between the insides and outsides of the cell. When a neuron stimulates muscle fiber, it depolarizes as the signal goes along it. Depolarization of one fiber, causes depolarization of all fibers in neighbourhood, what generates small electric field near each muscle fiber, [9], which can be detected by EMG sensors. Bigger contraction - means bigger number of cells depolarized, what results in stronger feedback on EMG

3.2 Muscles involved in arm movement

As surface EMG works best, when sensor electrodes are placed on top of the muscle, we need to know which muscles are involved in hand movement. Basically kinematic model of the human arm looks like the following and has 7 degrees of freedom: 20 different muscles of different sizes actuate an overall arm [10]. But we

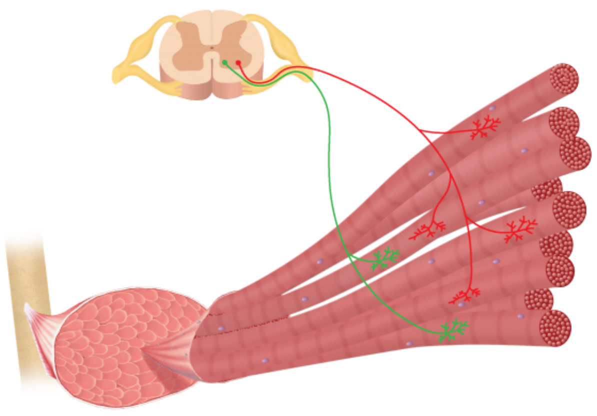

FIGURE 3.1: Motor unit diagram
(Image drawn by BYU-I student Nate Shoemaker, Spring 2016)

will be interested only in the biggest of them. For the 1 experiment described in Chapter 6, we only interested in muscles that actuate elbow joint ($Q4$ DOF of the arm at Fig. 3.2). They are **biceps brahii - triceps brahii**, shown at Fig. 3.3.

In experiment 2, we interested in the shoulder and elbow joints ($Q1, Q2, Q3$), so there is also no need to place sensors on each of all 20 muscles because we can distinguish between different muscles in the same group, based on data from sensors, in detail, this is described in Chapter 5. So we choose the following pair of muscles and corresponding joints:

- **Pectoralis majoris - Deltoid** - muscles main actuators of the shoulder joint ($Q2$ DOF of the arm at Fig. 3.2),
- **Pectoralis majoris - Latissimus dorsi** - muscles main actuators, used in moving elbow front and back (like uppercut punch or elbowing someone behind you) shoulder joint ($Q1$ and $Q2$ DOF of arm at Fig. 3.2)
- **Trapezius - Pectoralis majoris** - muscles main actuator, used in moving elbow front and back, in T-pose position of the arm in the shoulder joint ($Q1$ DOF of the arm at Fig. 3.2)

All of the above muscles can be seen at Fig. 3.4.

3.3 Hardware definitions

In the next chapters, some hardware definitions will be used, so here it will be briefly described:

3.3.1 DMA

DMA is a feature of modern MCU and CPU, which allow us to transfer data from and to the external device in the background while not loading the processor with this task

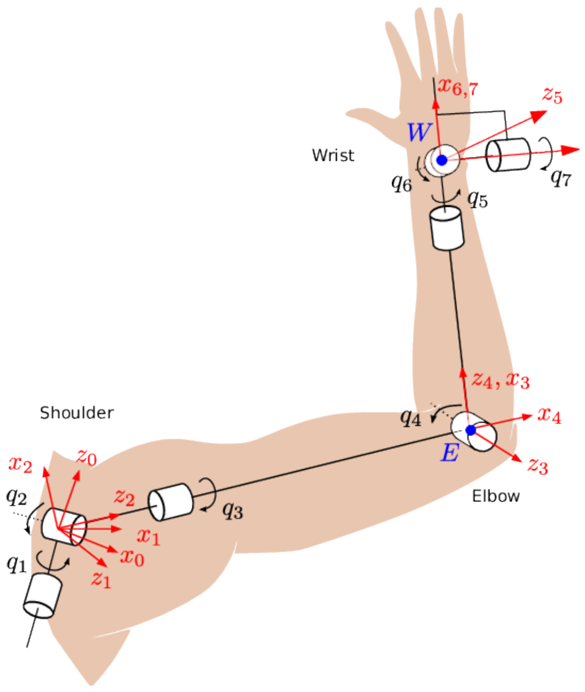

FIGURE 3.2: Human arm kinematics model
(Image from [18])

3.3.2 ADC

ADC is the piece of hardware that allows converting analog data to its digital representation. STM32 uses specific type of ADC — SAR (Successive Approximation Register), what means that conversion is performed in several steps and uses capacitors of different capacitance. First, capacitors are charged for some amount of time (usually called sampling cycles), which means - more sample cycles we give to ADC, the more charge will be collected in capacitors, so more accurate measurements can be performed. Next step — capacitors one by one are connected to the comparator pin, which checks if this specific capacitor is fully charged, so we have C sampling cycles plus 12 cycles (one per each bit, so as we use 12 bit ADC - we have 12 cycles) to read data from ADC

3.4 ROS

For this thesis, we must develop a system to collect data for our experiments, so we chose ROS as a framework because it is modular and easy to use. The main information to understand here, ROS solutions consists of two main components:

Node
A small independent program, written on C++ or Python, like object detection, data acquiring from the sensor, etc.

Topic
Channel through which **nodes** are communicating. **Nodes** can subscribe on **topics** to obtain info from them, or publish to **topics** to push info into them. If few **nodes** subscribes to the same topic, each of them will get the same set of messages.

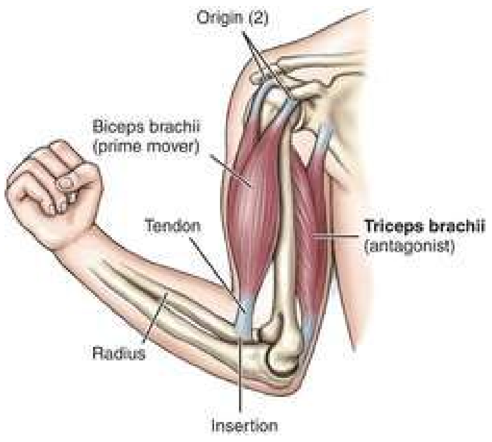

FIGURE 3.3: Muscles involved in elbow movement
(Image from [15])

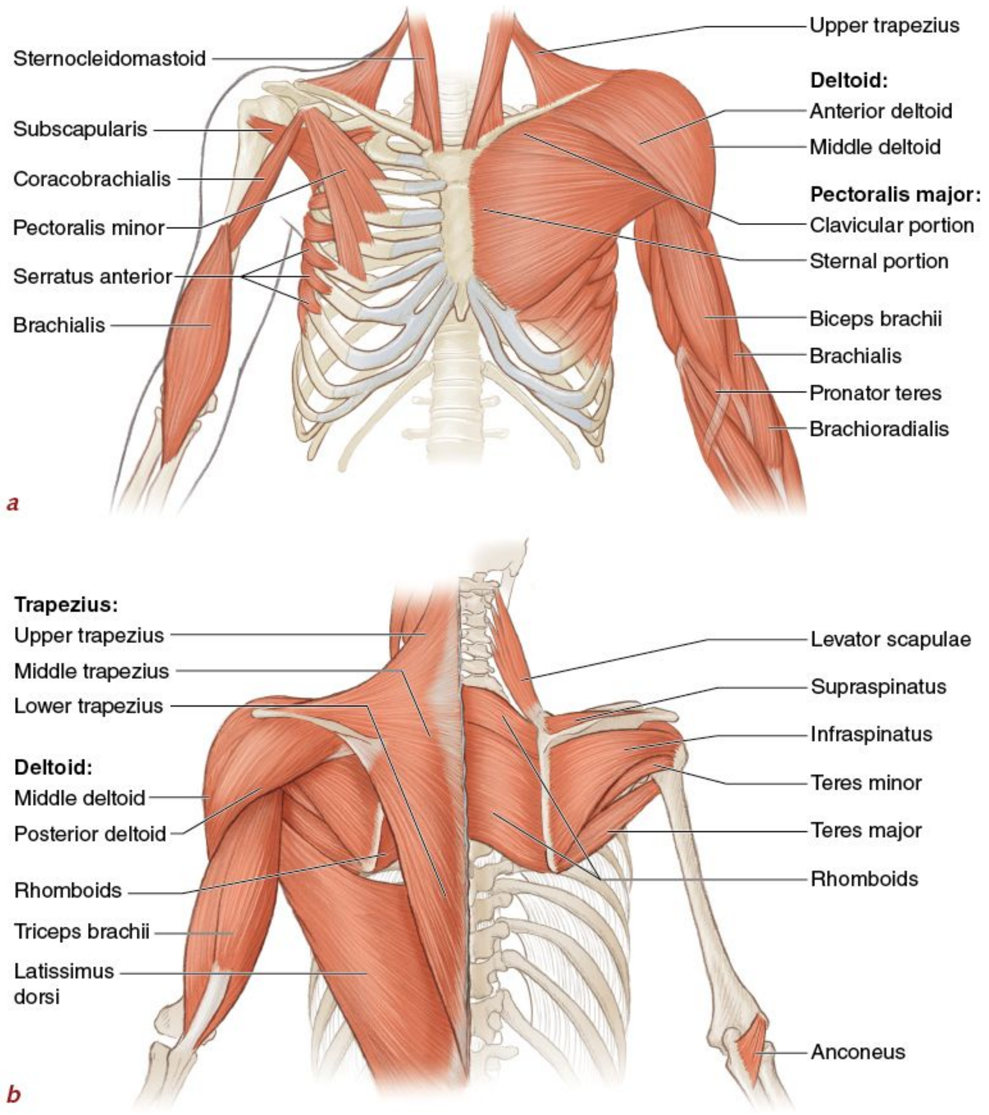

FIGURE 3.4: Muscles involved in elbow movement
(Image from [13])

Chapter 4

System overview

During research, no datasets for the continuous motion were found, so we decided to collect them by ourselves. Therefore data collection system should be created both in hardware and software.

4.1 Hardware

The hardware collection system consists of MCU (STM 32) and an array of 10 home-made EMG sensors. STM 32 was chosen due to its full-speed USB interface (12MB/s), 12 bit ADC, and high-frequency CPU. What about EMG sensors, was decided to assemble them by ourselves. In detail, the scheme will be described in the next section. Moreover, surface EMG sensors were chosen, as they don't require sterile electrodes which are inserted into muscle tissue.

4.1.1 EMG sensor

Basically, any EMG sensor consists of 4 parts [3]:

- **Pre-amplifier** — Voltage difference between electrodes placed on the muscle is very small, so we need to amplify it using a low noise difference amplifier
- **Amplification** — Next, we need to amplify the output of the previous amplifier even more, to be able to process it further
- **Rectification** — As we are using positive logic in our STM32 MCU, we need to rectify our positive/negative voltage, so we use a full-bridge rectifier to turn negative voltage into positive
- **Smoothing and amplification** — In the last part of the EMG circuit, we filter our signal with a low-pass filter to get rid of humps in the rectifier output to produce a smooth input signal for MCU. Finally, we amplify the signal even more, using an amplifier, capable of manual gain configuration

4.1.2 MCU

Finally, the output of the sensors is processed by MCU. We use a built-in 12 bit ADC of STM32F407 MCU. Due to the ADC structure, described in Chapter 3 we need to specify how many clock cycles ADC performs sampling of data, where bigger cycles count means more accurate results. Therefore, when MCU clock frequency is 32 MHz, and we want 1 kHz sampling frequency for our system, we can have total

$$C = \frac{32\,\text{MHz}}{1\,\text{kHz}} - 12 = 31988 \tag{4.1}$$

sampling cycles for all 10 channels. We also use DMA to quickly move ADC results as soon as they are done to MCU memory without CPU usage.

4.2 Software

For the software part of the system, we decided to use ROS due to its modularity and convenient API for Python and C++. Below we can see the diagram of ROS nodes.

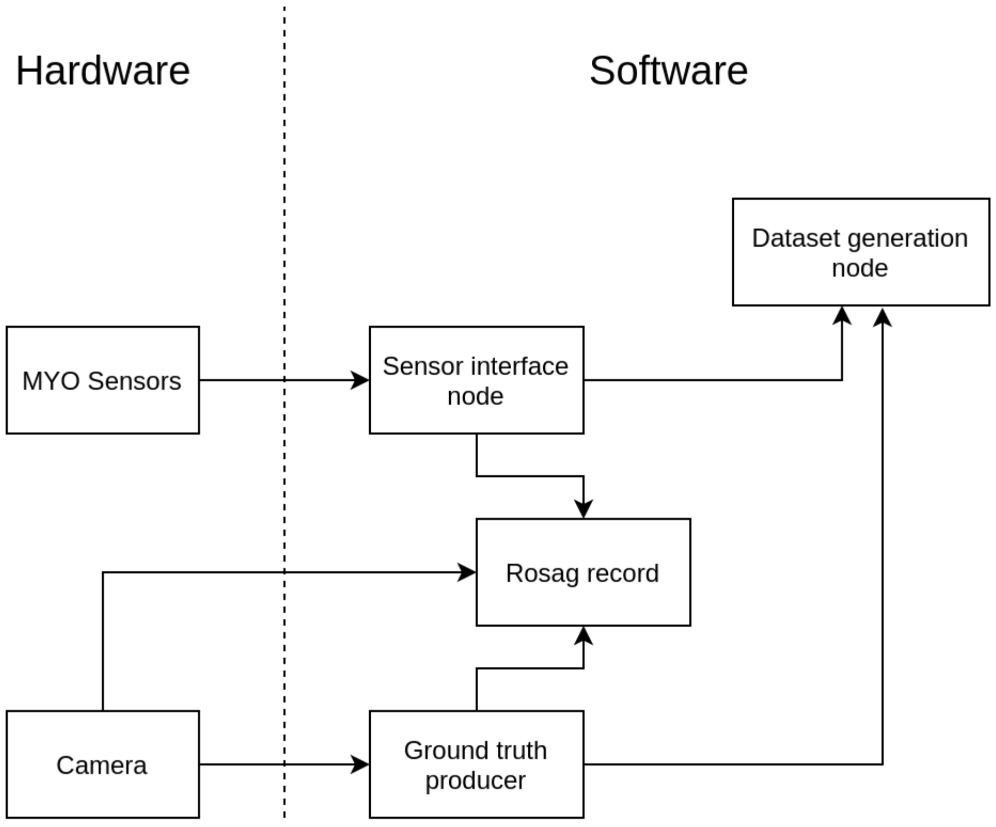

FIGURE 4.1: System diagram

- **Sensor interface node** — node which is interfacing with sensor array converting it from raw bytes in serial to ROS messages
- **Ground truth producer** — node which will generate ground-truth (real hand positions, which would act as labels for EMG data, used to train neural networks) for our dataset, which can be replaced with different detector nodes. In Chapter 7, the first experiment, this node is replaced with a simple arm position detection node. In the second experiment - the DNN mocap node is used. In detail, these nodes will be described later.
- **Dataset generation node** — this node aggregates and synchronizes data from two previous nodes and saves it as a CSV dataset
- **Rosbag record** — this node actually is a ROS feature that allows us to record topic data into ROS-bag files. In our case, we record camera images at 5 Hz rate, for reference, when we are investigating data, and all data from the first

two nodes, so we can always recover data in case of **Dataset generation node** fail.

4.2.1 Ground truth producer

Our system architecture is modular, due to the usage of ROS. So we have a possibility to replace nodes as long as they implement the same interface. We used this feature to replace Mocap neural network node with different nodes for different experiments. For example, in the first experiment (proof of concept), as ground truth we need only elbow angle, so we use a simple arm position estimation node, which gives us only a 1D pose of the elbow joint. While for the second experiment, we use a full-body motion capture neural network to get shoulder angles in 3 planes.

Simple arm position estimator

This position estimator is implemented using classic computer vision algorithms. As input, it takes RGBD images (RGB image with additional depth channel), converts RGB image to HSV format, thresholds it to find 5 markers placed on the arm like this — Fig. 4.2.

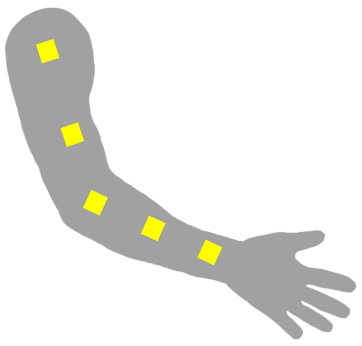

FIGURE 4.2: Marker positions on hand

After that algorithm uses a pinhole camera model and depth channel of the image to find positions of markers in 3D space. Next, we try to find marker on elbow joint, and two lines which represents forearm and shoulder. To do this, we try to find best fitting lines for every 3 marker combination from 5 markers. Therefore we select two lines, which fits best and the intersection between them, what gives us all required information (Fig. 4.3).

Finally, find a plane through them and angle. This angle is further used as ground truth data for the dataset.

DNN motion capture

This node is based on the open-source realization of DNN motion capture solution — BlazePose [2], which in real-time can detect and track key points of the human body in 3D. We added a ROS interface for this solution and integrated it into our

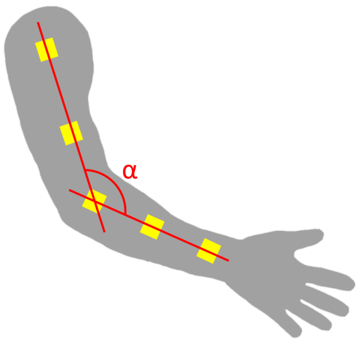

FIGURE 4.3: Estimated lines on hand

system. Also, as we need only data about one arm, so we filter out all excessive information.

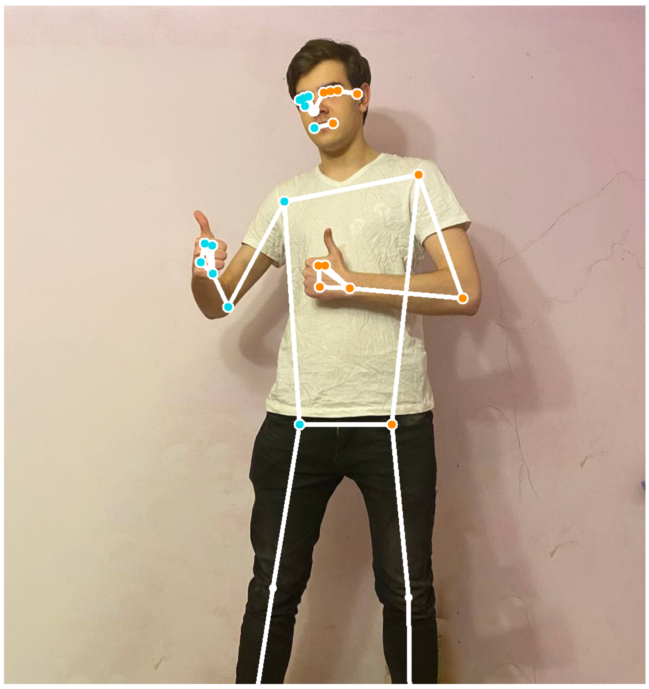

FIGURE 4.4: Motion capture example

Chapter 5

Data overview

5.1 Dataset description

For each of two experiments, we will be obtaining EMG data with our hand-made sensor array and ground truth data obtained from camera images, as described in 4. So, the dataset for each experiment looks like following Fig. 5.1.

One of the problems with our dataset is that the ground truth data is not synchronized with EMG data (processing of the image takes more time than obtaining data from the sensor, also, as we use neural network pose estimation, sometimes body track is lost for few frames, so we need to fill in gaps in data). So in our system, we used timestamps for each message sent over topics to synchronize them in one dataset later. Another problem is a different frequency of data (approximately 30 Hz for ground truth data and 1 kHz for EMG data) what is illustrated on Fig. 5.1, where points represents moment of time when EMG/Ground truth data were received.

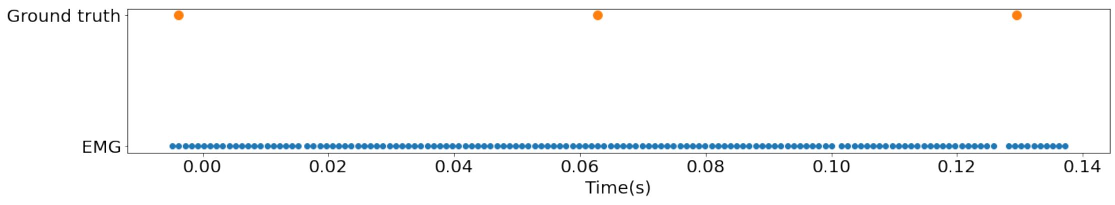

FIGURE 5.1: Data frequency demonstration

So, first of all, we perform data synchronization and interpolation to make one coherent dataset. Finally, we get the following result — Fig. 5.2, which we use as the dataset for further pre-processing and neural networks training.

5.2 Signal pre-processing

As can be seen at 5.3, our signal is slightly noisy, but this noise is not periodic, which can be due to the high-frequency (>1 kHz) noise, or ADC noise. Also, we don't have

TABLE 5.1: Dataset schema

Timestampp	ADC 1	...	ADC N	Joint 1 angle	...	Joint N angle
1620595463	2075	...	856	0.5	...	1.55
...	...	...	...	...	...	...

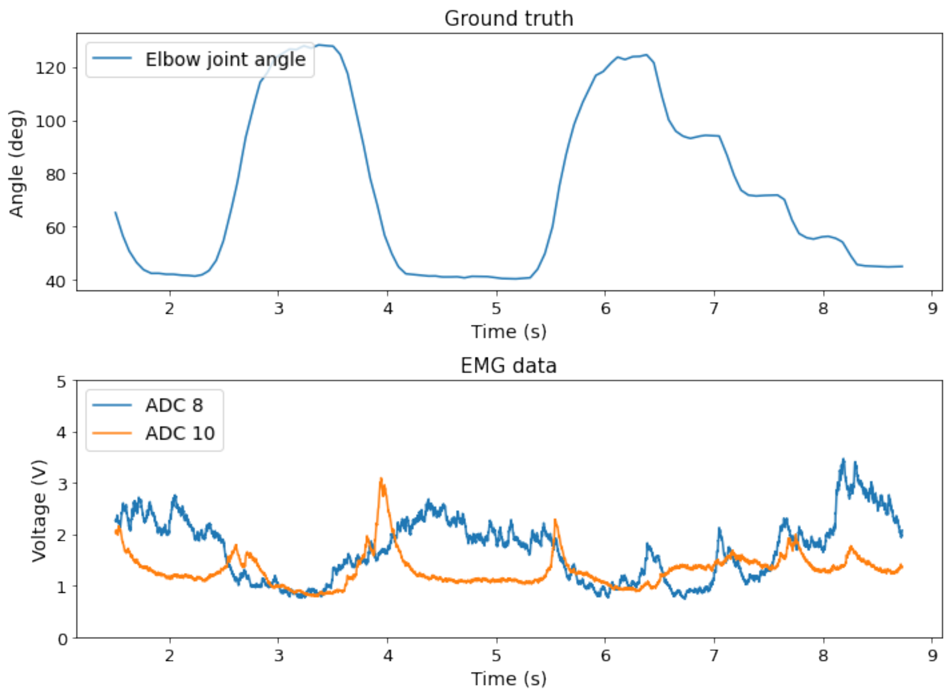

FIGURE 5.2: Dataset example (Experiment 1)

50 Hz noise of AC power grid, as we are using batteries to power analog part of the scheme, as well as we have reference electrode in sensor scheme, which also reduces noise to level, which is beyond detection threshold of our system.

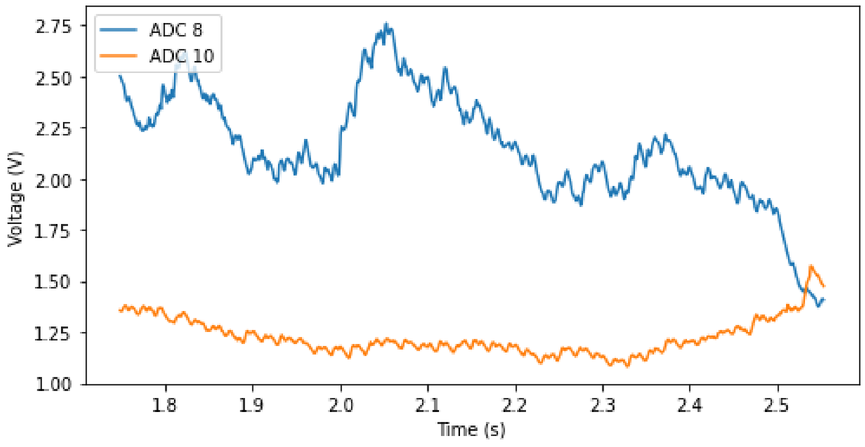

FIGURE 5.3: Raw signal example (Experiment 1)

As our signal is already rectified on the hardware level, there is no need to do it by ourselves, so our processing starts with filtering noise by using RMS. We can use it as it correlates with signal power [14].

Also, for some cases in Chapter 6 we used additional extracted features — Hjorth parameters — statistical features of signal in the time domain, originally developed for EEG analysis:

- Hjorth Activity — mean power of a signal, its activity

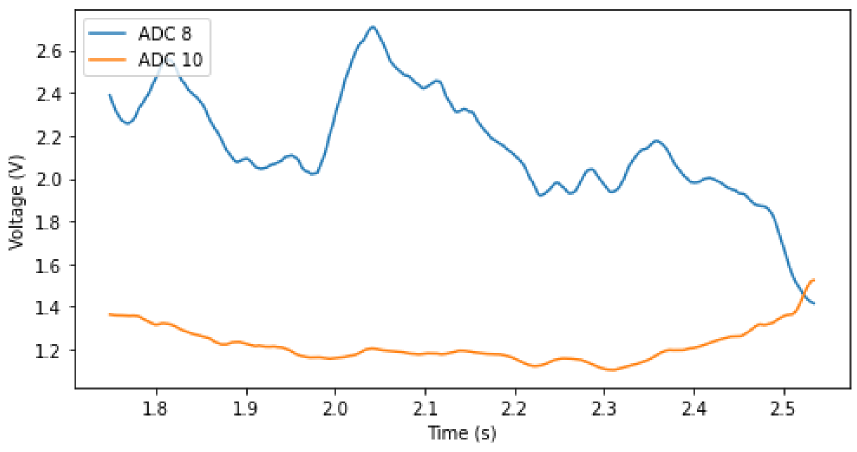

FIGURE 5.4: Filtered signal example (Experiment 1)

- Hjorth Mobility — mean frequency of the signal
- Hjorth Complexity — estimate of signal bandwidth

5.3 Cropping dataset to batches

In Chapter 6 we use neural networks for data analysis, so we need to split the dataset into batches. Moreover, we want to use a trained neural network for real-time inference, so we choose a sliding window to cut the dataset on batches.

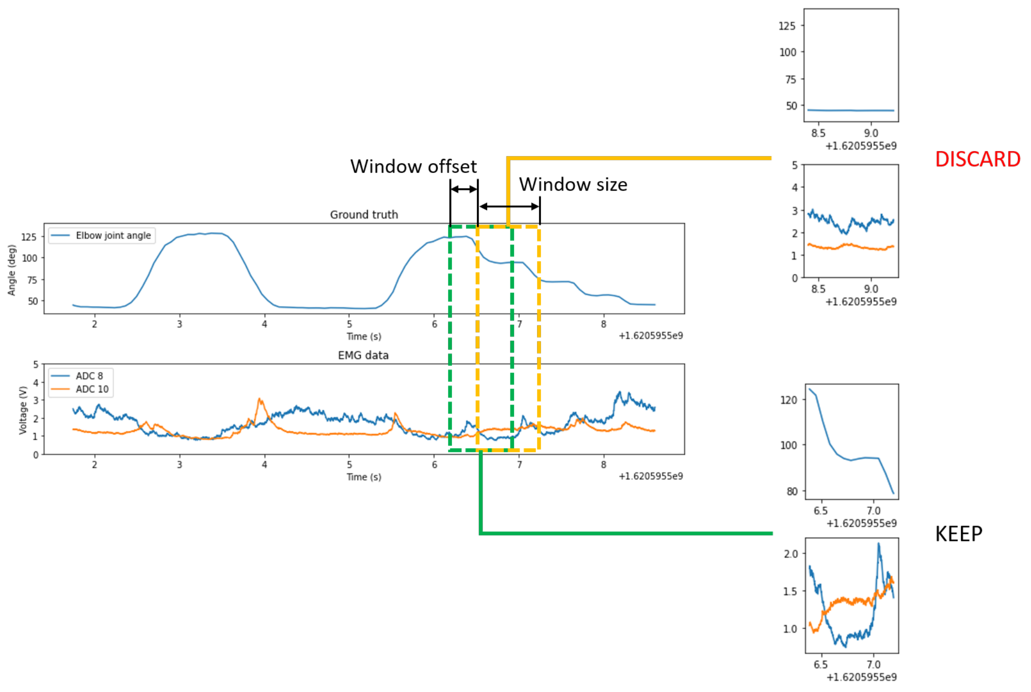

FIGURE 5.5: Cutting dataset on batches

Also, from the dataset, we see that number of "static" batches (the part of the dataset where there is no movement) will be bigger than batches with some useful information, so we detect them by analyzing the absolute change of total joint position angle, and if it is smaller than the threshold - put it into other lists, from which random part will be discarded, to balance our dataset. Also in Chapter 6 we will choose the size of the window and its shift to be good enough for small latency on inference, and small redundancy, for training data generation to not overfit neural networks.

5.4 Distinguishing between different movements from one signal

In this data sample, we can see data from one EMG sensor placed on the forearm, for example, as the forearm has a large amount of overlapping muscles. In this small experiment, we perform two types of moves, repeated twice, as shown at Fig. 5.6

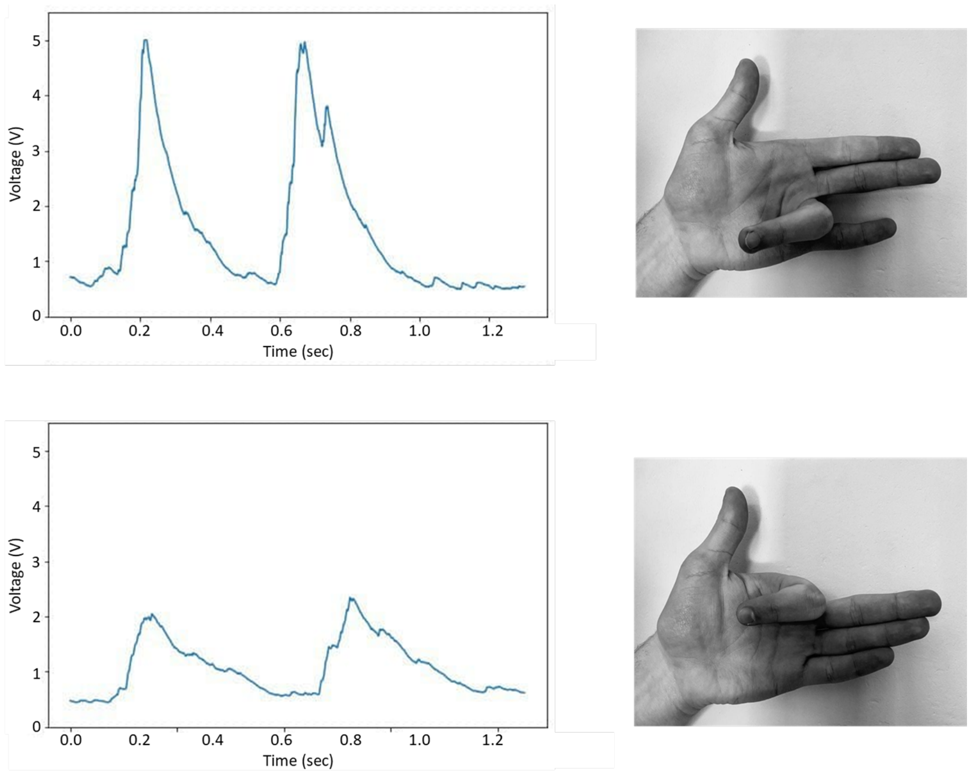

FIGURE 5.6: Different movement types, from same signal sample

As can be seen at Fig. 5.6, there is a clear difference between the signal from the single sensor for different gestures due to the fact that surface EMG captures total potential difference from all motor units under the sensor, and different gestures need different contraction level from muscles, what results in a difference in EMG signal amplitude.

Chapter 6

Methodology

In this work, two experiments will be conducted - first with joint with only 1 DOF — elbow, to find the best approach for EMG analysis, which will work for continuous motion and second — with shoulder joint, which is more complicated as it has 3 DOF, actuated with overlapping muscles. So the idea is - check if our approach is feasible in experiment 1 and check and improve it in experiment 2. From Chapter 2 we know few approaches for EMG data analysis, like feature engineering or direct passing signal to the neural network. In our case, we also will experiment with different neural network output — we have the assumption that the result of our neural network should be not a position of the joints but the velocity at a given point of time.

6.1 Validation metrics

To find the best approach of analyzing EMG data, we need some metrics by which we are comparing results. Basically, we are solving regression tasks, so we cant use the accuracy metric. Therefore we are calculating an error between prediction and actual data, trying to minimize it. Moreover, our regression is nonlinear, so we cannot use some metrics, like R^2.

So we stopped at following metrices.

- MAE — Mean Absolute Error - this metric shows difference between true and predicted data, disregard it's sign, what allows us to compare performance between proposed models.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|x_i|$$

- RMSE — root mean squared error — this metric works quite the same as previous one, but due to the power of 2 in formula, outliers have bigger effect on overall score, what allows us to better compare how many outliers every model produces.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$$

6.2 Experiment 1

6.2.1 Experiment description

In this experiment, we gathered data from **bicep brahii** and **tricep brahii**, which play the main role in elbow joint actuation. Overall we have 30 min of data, with the following moves conducted:

- full bending and straightening of the elbow joint, type 1 in Fig. 6.1
- bending and straightening from fully relaxed to half bend and from half bend to fully bend, type 2 in Fig. 6.1
- bending on half of the overall magnitude - starting from first quarter of cycle to third quarter of a cycle, type 3 in Fig. 6.1
- straightening and bending with a step of 10°, type 4 in Fig. 6.1

During all these tests, we were trying to maintain a constant movement speed. In further researches, we will try to train the system to analyze movement at different speeds. But for now, it will add an additional complexity to our system.

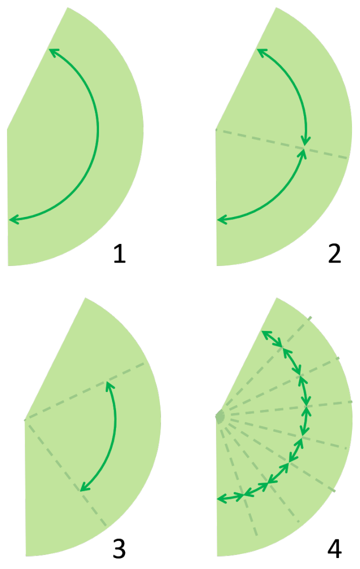

FIGURE 6.1: Experiment 1 moves (green zone means possible forearm movement)

6.2.2 Methodology

In Chapter 3 we described existing approaches of EMG signal recognition and concluded that we would use neural networks for analyzing EMG data. We tried different architectures of the neural networks, as well as different dataset generating methods. As was said in Chapter 6 we used a sliding window for generating training data, so for each NN architecture, we experimented with different window size and window shift, but the sum of window shift and window size was always less than 300, to keep latency quite small, so the system is considered near-realtime. For the labeling of data, there are two different approaches. We may want NN to produce an exact angle (further marked as α), on which hand joint is rotated in the specific moment, but in some cases, it wouldn't work well because for some states of our hand, EMG data looks identically, for example fully bend and almost fully bend arm in elbow joint, configuration of muscles activity in this exact moment is the same, but hand position is different, so we also considered using "angle change over window" (further marked as $\Delta\alpha$) as a label for data. In each experiment, after training, we perform a test on data, previously hasn't seen by the neural network (same part of the signal for each neural network), further analysis of the output can be seen in Chapter 7

6.2.3 ANN

The first approach is rather naive. We tried to use simple neural network architecture — Fig. A.1 with data preprocessing. In some approaches [20] Hjorth parameters were used to predict discrete gestures, so we decided to use them for our data. We feed to neural network 3 Hjorth parameters, as well as some statistical features, like mean over the window and standard deviation for each of 2 channels along with our raw EMG data, to provide neural network all available information. After some finetuning, we got the following architecture — Fig. A.1, which was trained for two types of output - angle change over window time ($\Delta\alpha$) and joint angle itself (α), which gives us the following prediction accuracy — Tab. 6.1 and following prediction for the test data sample — Fig. 7.1.

TABLE 6.1: ANN prediction accuracy

Metric	Angle change over window	Raw angle
RMSE	0.012	0.204
MAE	0.016	0.168

6.2.4 RNN

RNN — a type of neural network, that have small memory blocks inside, so it remembers previous calculation, therefore perfectly suits for time series prediction or even EMG data analysis. For previous neural network architecture, we extracted some features, but for RNN, we use raw signal, with a small amount of filtering, described in Chapter 5. The architecture of our recurrent network is shown in Fig. A.2 What produces the following accuracy for $\Delta\alpha$ and α prediction — Tab. 6.2 and the following prediction for the test data sample — Fig. 7.2

TABLE 6.2: RNN prediction accuracy

Metric	Angle change over window	Raw angle
RMSE	0.077	0.078
MAE	0.052	0.051

6.2.5 RNN with previous prediction input

As can be seen in the previous RNN prediction, it's slightly noisy, so we decided to give recurrent neural network information about the previous state of the hand during the previous window (200 ms) — Fig. A.3. For training — ground truth elbow angle was used, when for validation, previous predictions of the neural network itself were used. Output can be seen in Fig. 7.3, and accuracy in Tab. 6.3

TABLE 6.3: Enhanced RNN prediction accuracy

Metric	Angle change over window	Raw angle
RMSE	0.109	0.077
MAE	0.086	0.056

6.2.6 CNN

We tried to use pure CNN on our data, as described in related works, but it didn't work out well for our data and approach, so we experimented with the combination of 1D Convolution and LSTM architecture — Fig. A.4. The idea behind this architecture is to allow 1D Convolution to generate features for LSTM. This approach produces the following result on test data — Fig. 7.4, and its accuracy is following — Tab. 6.4

TABLE 6.4: CNN prediction accuracy

Metric	Angle change over window	Raw angle
RMSE	0.072	0.086
MAE	0.047	0.057

6.2.7 Combined approach

As can be seen from RNN and ANN section, both of these networks somehow analyze our data. RNN is better, while ANN was slightly worse, but in different parts of prediction, as can be seen from Fig. 7.2 and Fig. 7.1. So we decided to combine both of them in one architecture — Fig. A.5, with two inputs - temporal for the RNN part of the neural network - and Hjorth parameters for the DNN part. What gives us the following accuracy — Tab. 6.5 and following sample prediction — Fig. 7.5

TABLE 6.5: Combined approach prediction accuracy

Metric	Angle change over window	Raw angle
RMSE	0.079	0.085
MAE	0.052	0.056

As can be seen from figures: Fig. 7.5, Fig. 7.2 and Fig. 7.1 — combined network, in some cases, even slightly outperform both RNN and ANN, but overall its performance is worse.

6.2.8 Enhanced combined approach

As in the previously combined architecture, we also decided to combine ANN with RNN with the previous prediction input result. So we combined those architectures, getting the following — Fig. A.6. What gives us the following accuracy — Tab. 6.6 and following sample prediction — Fig. 7.6

TABLE 6.6: Combined approach prediction accuracy with previous prediction input

Metric	Angle change over window	Raw angle
RMSE	0.106	0.107
MAE	0.080	0.079

6.3 Experiment 2

6.3.1 Experiment description

In this experiment, we gathered data from **Pectoralis majoris**, **Deltoid**, **Latissimus dorsi** and **Trapezius**, which are main muscles-actuators of elbow joint, and move it in all 3 DOF. Overall we gathered 45 min of data, with different movements of the elbow joint, with the following moves conducted:

- elbow is in most far back position, moves up end down, then shifts slightly to the front, moving up and down, repeats, type 1 in Fig. 6.2
- same as above, but repeating part of the move is performed horizontally, type 2 in Fig. 6.2
- spiral moves of the elbow from top to bottom and from left to right
- diagonal moves of the elbow
- same 4 moves, but with rotating elbow along elbow axis
- combination of all of the above moves, for test part of dataset

6.3.2 Methodology

From experiment 1 we know, that the best approach for 1 DOF joint angle estimation is pure LSTM for angle change over window prediction and RNN with previous prediction input for raw angle prediction Now, we will use those approaches for detecting complex movement signal (4 muscles data as input, 3 angles as output). Moreover, we slightly tuned our architecture to show better performance on 4 input data streams, RNN architecture, for "angle change over time" prediction is shown in Fig. A.7 and Enhanced RNN architecture, for angle prediction is shown in Fig. A.8.

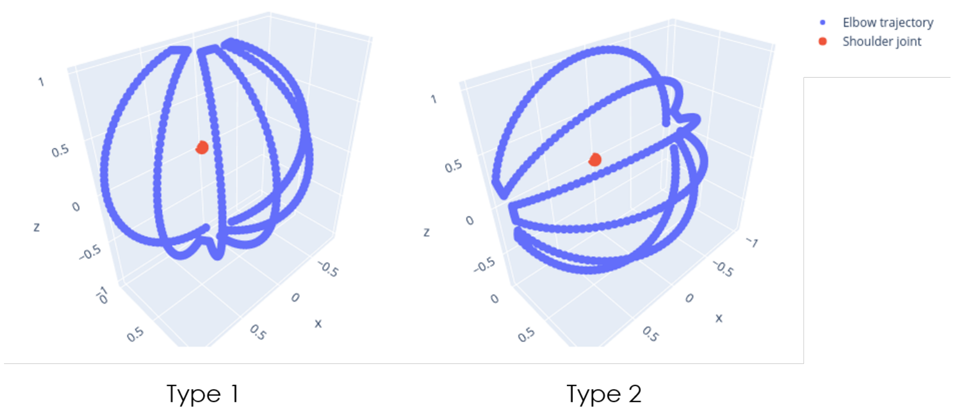

FIGURE 6.2: Experiment 2 moves (blue dots means elbow trajectory during move)

TABLE 6.7: Experiment 2 RNN accuracy with previous prediction input for raw angle prediction

Metric	X	Y	Z	Total
RMSE	0.129	0.135	0.306	0.129
MAE	0.079	0.12	0.28	0.102

Overall we get the following accuracy for angle prediction — Tab. 6.7 with following prediction, for the test data sample — Fig. 7.7:

Speaking about prediction angle change over window we get following accuracy — Tab. 6.8, with following prediction — Fig. 7.8 on the test data sample

TABLE 6.8: Experiment 2 RNN accuracy for angle change over window prediction

Metric	X	Y	Z	Total
RMSE	0.097	0.062	0.091	0.084
MAE	0.077	0.049	0.064	0.063

Chapter 7

Experiment results

7.1 Experiment results

During our experiment, we tested 5 architectures of neural networks for predicting continuous arm pose from EMG data. So in this chapter, we will compare them in details and analyze differences and possible improvements.

Overall, all of the described architectures were trained on the same dataset, with training parameters fine-tuned to get the best possible result, after that, it was tested on the test part of the dataset — the part which the neural network has never seen. All tests were performed to match real use-case. With common neural network architecture, we just use test part of the dataset. But in tests of neural network, with old predictions input (Enhanced RNN) we cannot use test part of dataset as it is, because inside of it, as an "old predictions input" we have previous ground truth data, but in real life, we doesn't have it, and it will be replaced with previous prediction of neural network, so in our tests, we also performed inference one-by-one sample and altering data from dataset with previous prediction of neural network. Results and analysis of those tests can be seen in the sections below.

7.1.1 ANN

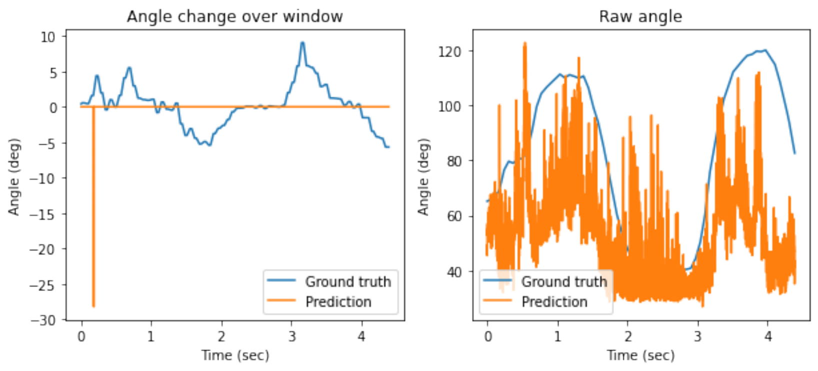

FIGURE 7.1: ANN prediction sample

From these results, we can see that multi-layer perceptron, trained on extracted features, can predict raw angle from EMG data, but do this very jittery, what of course, can be filtered and post-processed but will not give us the best possible accuracy. What's about predicting "angle change over the window" with the simple ANN, from Fig. 7.1, we can see that neural network was not learning on this type of label at all, what results in the simple median line, instead of propper result.

7.1.2 RNN

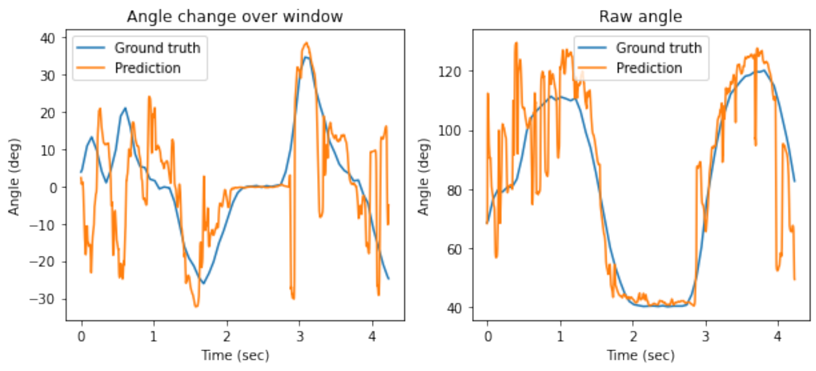

FIGURE 7.2: RNN prediction sample

Speaking about recurrent neural network, we can see that it predicted raw angle quite good, despite the fact of jittering, which can be seen on the first ascending part of the graphics. This can be due to the relatively small dataset (only 30 min of data). Despite that, we can still see the trend of the prediction, so with some effort and filtering, we can even get a quite good result with LSTM approach. Speaking about angle change over window — RNN prediction also works out better than ANN, but still has jittering issue in 0 s-1.5 s part of the test data, which means that indeed, we have an issue with dataset size, which can be solved if we gather more data in more conditions (more different arm poses, different humans), what will give us the more generalized model, which has seen more different cases, therefore will be better in predicting it.

7.1.3 RNN with previous prediction input

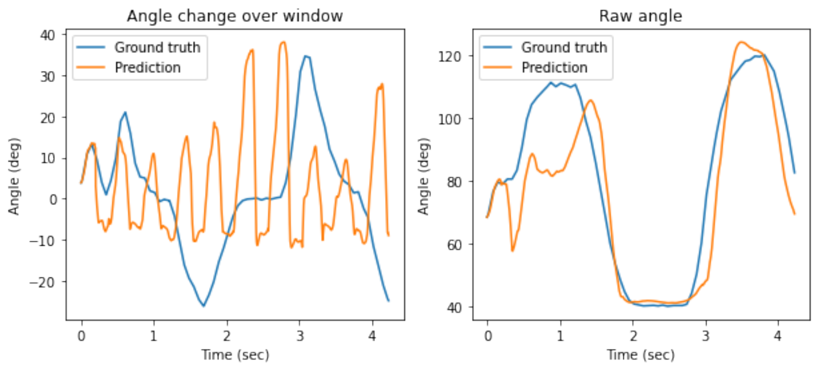

FIGURE 7.3: Enhanced RNN prediction sample

In this architecture, using previous predictions in combination with EMG data, we get rid of jittering, our prediction for raw angle become smooth and very close to the ground truth, except mentioned earlier 0 s-1.5 s part of the prediction. Speaking about angle change over the window - we see artifacts — repeating patterns of peaks, which. as we assumed. it was from the too big size of the previous prediction buffer, so RNN learned features that are too far in the past, but even after tuning its size, we still didn't get rid of it, so for the Enhanced RNN case, only raw angle prediction is considered as a viable variant.

7.1.4 CNN

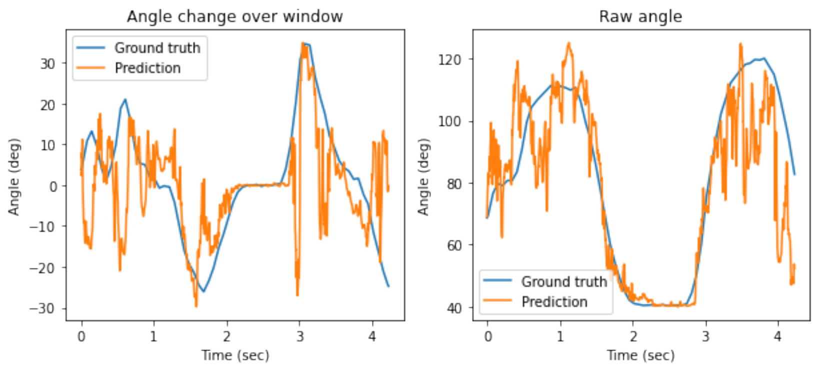

FIGURE 7.4: CNN prediction sample

As can be seen from Fig. 7.4, its output is quite the same as RNN results (Tab. 7.2). Same jittering and quite the same accuracy as can be seen in Tab. 7.1. So using Convolution for feature generation does not bring a big improvement to the detection accuracy.

7.1.5 ANN and RNN combined

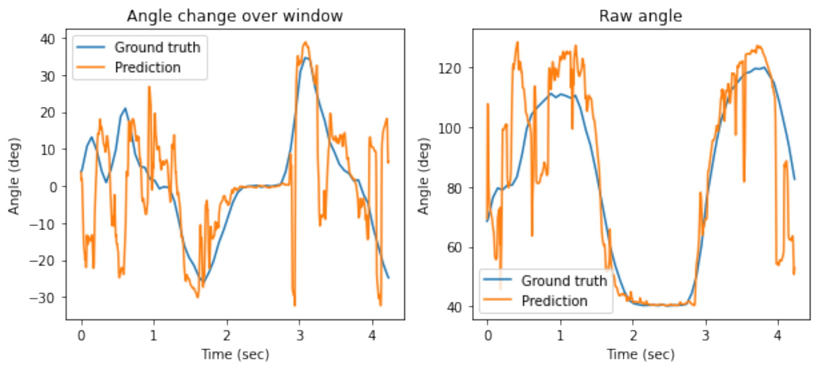

FIGURE 7.5: Combined ANN-RNN prediction sample

Our idea was that combination of different architectures with completely different approaches to processing should give us a better result, but as can be seen from the raw angle prediction case in Fig. 7.5, while in some parts (0 s-1.5 s of test data) this architecture performs better than RNN and ANN apart, but overall its performance is worse. What about angle over window prediction — generally, it performs almost the same as the RNN approach, which is expected, as the ANN approach doesn't learn anything in angle over window prediction case.

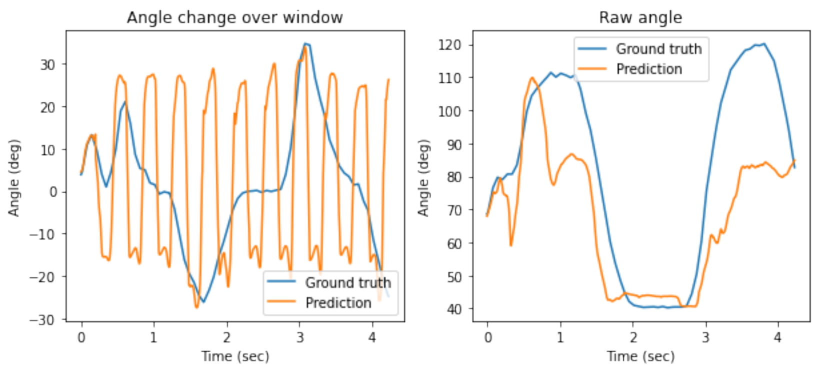

FIGURE 7.6: Combined ANN- enhanced RNN prediction sample

TABLE 7.1: Combined metrics of all NN architectures

NN architecture	Angle change over window		Raw angle	
	MAE	RMSE	MAE	RMSE
ANN	0.016	0.012	0.168	0.204
RNN	0.052	0.077	0.051	0.078
Enhanced RNN	0.086	0.109	0.056	0.077
CNN	0.047	0.072	0.057	0.086
Combined ANN-RNN	0.052	0.079	0.056	0.085
Combined ANN-Enhanced RNN	0.080	0.106	0.079	0.107

7.1.6 ANN and RNN combined, with previous prediction input

As in the previous case, our assumption was to combine the best from two worlds to get a better result, but the ANN part of the architecture, due to its jittering and big error, worsened the result of the RNN part.

7.1.7 Comparison

From the Tab. 7.1 we can form the following rating of NN architectures:

Angle change over the window:

- ANN — reason of such a big score for this architecture is due to its "fitting to the mean" as can be seen in Fig. 7.1, NN learned to produce constant number - mean of training data, which for our metrics looks like a good result, but visual analysis showed us, that this architecture is not capable of predicting angle change over window data.
- CNN — this architecture performs best on predicting angle change over a window from EMG data, based on our metrics and visual observation — Fig. 7.4. It is still slightly jittering, but we can clearly see the trend, which is quite aligned with ground truth data.
- RNN — this architecture was almost as good as the previous one, with slightly worse results, as can be seen in Fig. 7.2, visually prediction nature is quite similar to CNN one, due to its common architecture (LSTM blocks), but RNN is

slightly less robust, as it works with row data, not "preprocessed" with trained convolution layer.

What about raw angle prediction, rating is following:

- RNN/Enhanced RNN — that two architectures perform almost equally well in terms of metrics, but when we are talking about continuous motion recognition, we should consider jittering of NN output. In this case, Enhanced RNN is the clear leader as its output (Fig. 7.3) is much more stable than the output of RNN (Fig. 7.2)
- Combined ANN-RNN — this architecture, compared to the other two, was noticeably worse, as it has both jittering of RNN and worse accuracy of ANN (Fig. 7.5). But still better than other experiments

Overall we can make the conclusion that a better approach for continuous motion recognition is to use NN, which predicts actual joint angle, than angle change over a window. Also, for our task, the Enhanced RNN approach suits better, as its output is much smoother than RNN's.

7.1.8 ANN and RNN combined for experiment 2

Given the conclusion from the previous experiment, in this experiment, we also used Enhanced RNN for angle prediction, and simple RNN for "angle change over window" prediction, both neural networks training process were slightly tuned to suit new input data, which results in the following raw angle prediction from test data — Fig. 7.7

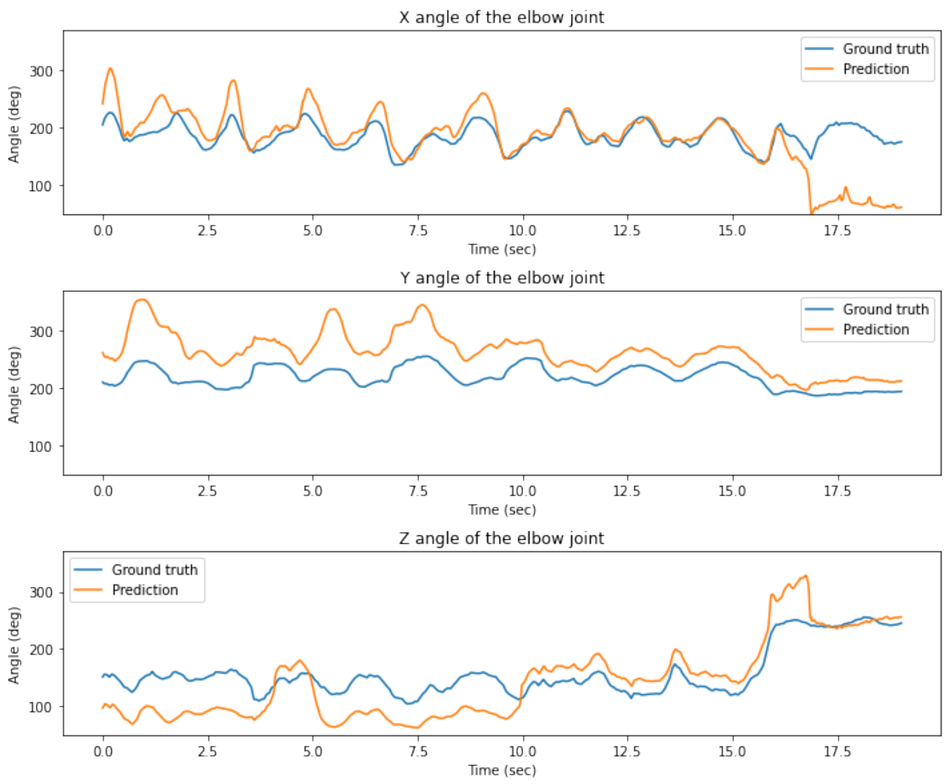

FIGURE 7.7: Experiment 2 Enhanced RNN prediction

As can be seen, from prediction sample, our approach works quite good even for complex joints - like elbow. Of course we can see some noise and mispredictions, which, as discussed above, can be due to small dataset with small ammount of different moves. So it can be eliminated by using bigger and more general dataset.

Spekaking about prediction of "angle change over window" — Fig. 7.8, from the metrics Tab. 6.8 and Tab. 6.7 we can see, that for complex joints, this approach slightly outperforms Enhanced RNN, but is much more noisier. What once again confirms our assumption, that for continious motion prediction Enhanced RNN architecture works best.

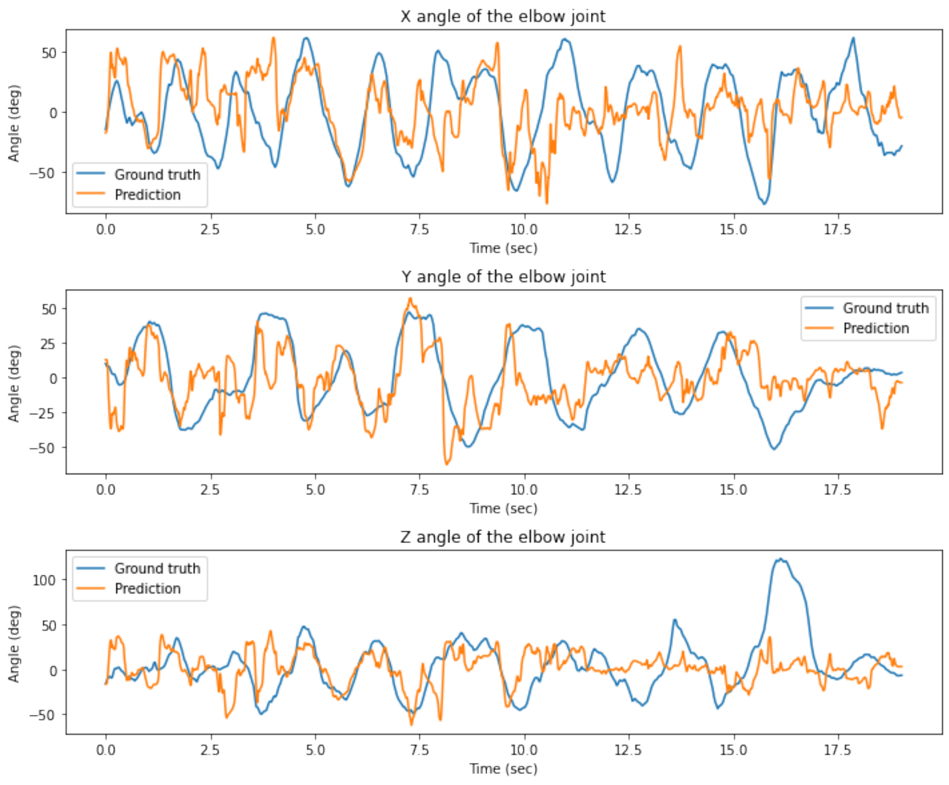

FIGURE 7.8: Experiment 2 RNN prediction of "angle change over window"

Chapter 8

Conclusion

In conclusion, during this work, a hardware and software platform for analyzing and gathering labeled EMG was created. It allows for everyone with proper hardware (which can be manufactured at home) to gather his own EMG dataset and therefore train his own regressor. In the main part of this work, were tested 5 different types of neural networks for 2 different types of output. Found best suiting neural network architecture for our task - recognition of continuous arm movement based on EMG data. Two models, for simple (like an elbow) and complex (like shoulder) joints, were trained and evaluated.

8.1 Model summary

For recognition of continuous movement, recurrent neural networks were used, which input is N channel EMG data, in windows of 200 ms as well as previous predictions of neural network (also for last 200 ms). Output is the angle/angles of the corresponding joint.

8.2 Future work

During this work, we faced a limitation connected with dataset size, as well as its diversity. So in future work, we will try to improve model accuracy by using a more general dataset, which will include people of different ages/sex/skin types, as well as many more different moves. Also, we will try to improve the stability of the model by using different filters based on knowledge of human arm kinematics.

Appendix A

Neural networks architectures

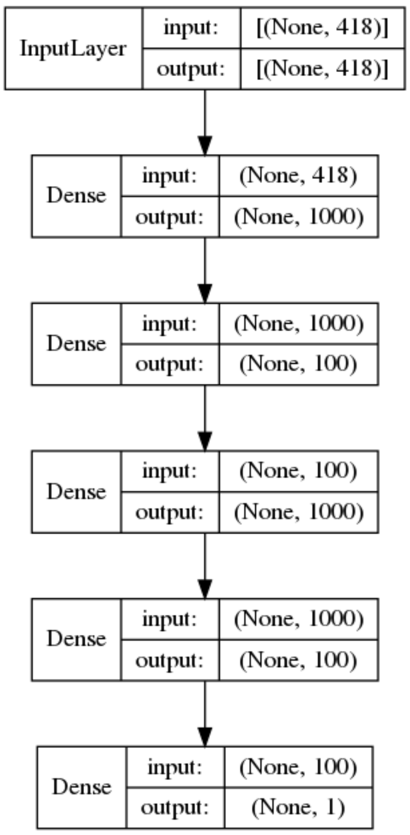

FIGURE A.1: Fully-connected neural network architecture

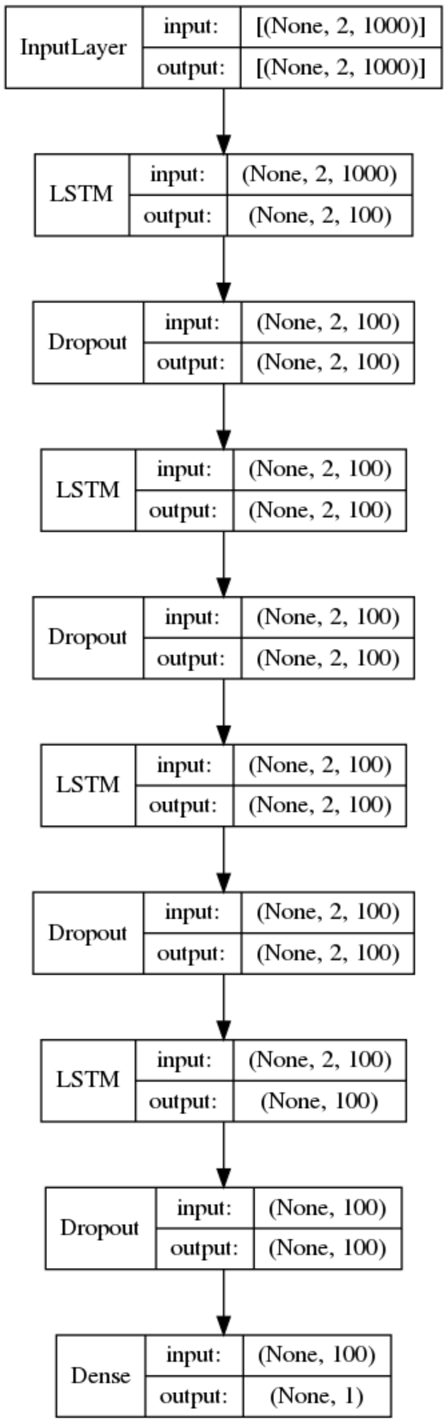

FIGURE A.2: Recurrent neural network architecture

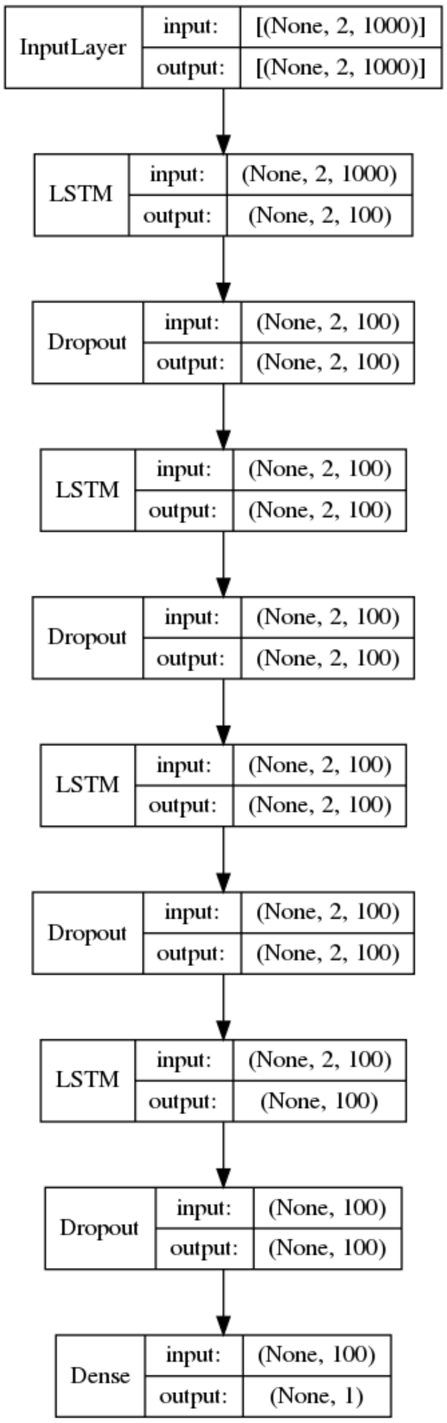

FIGURE A.3: Enhanced recurrent neural network architecture

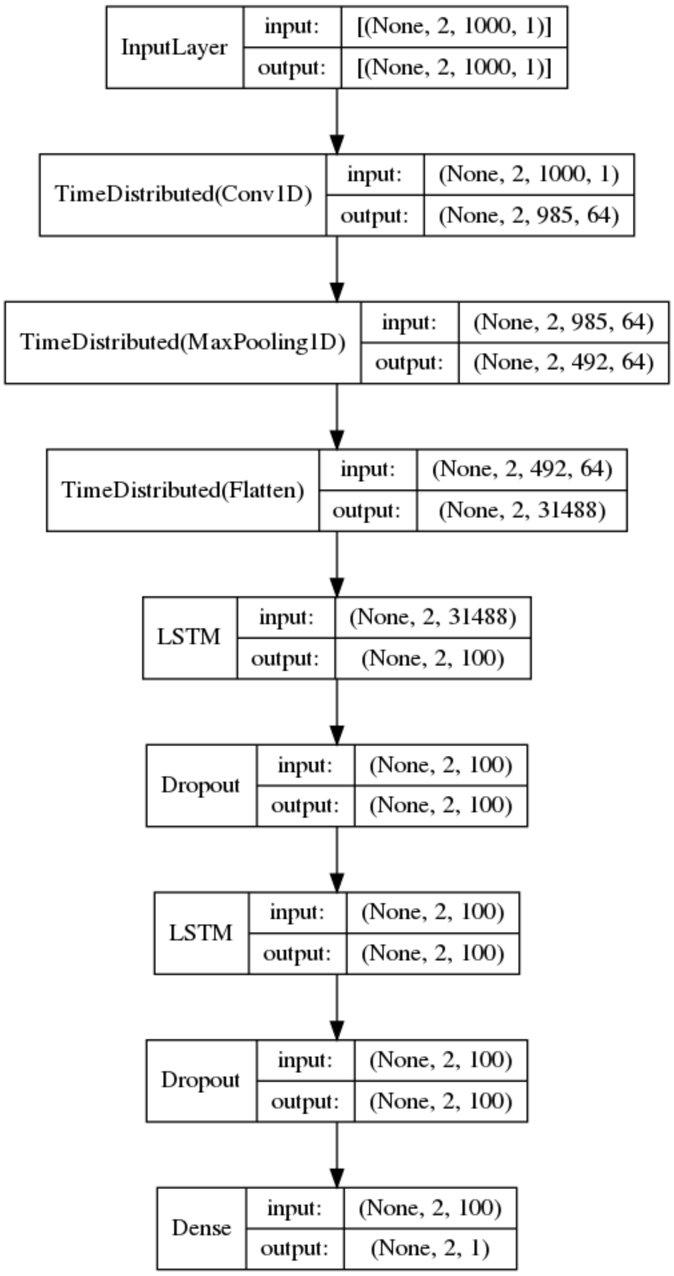

FIGURE A.4: Fully-connected neural network

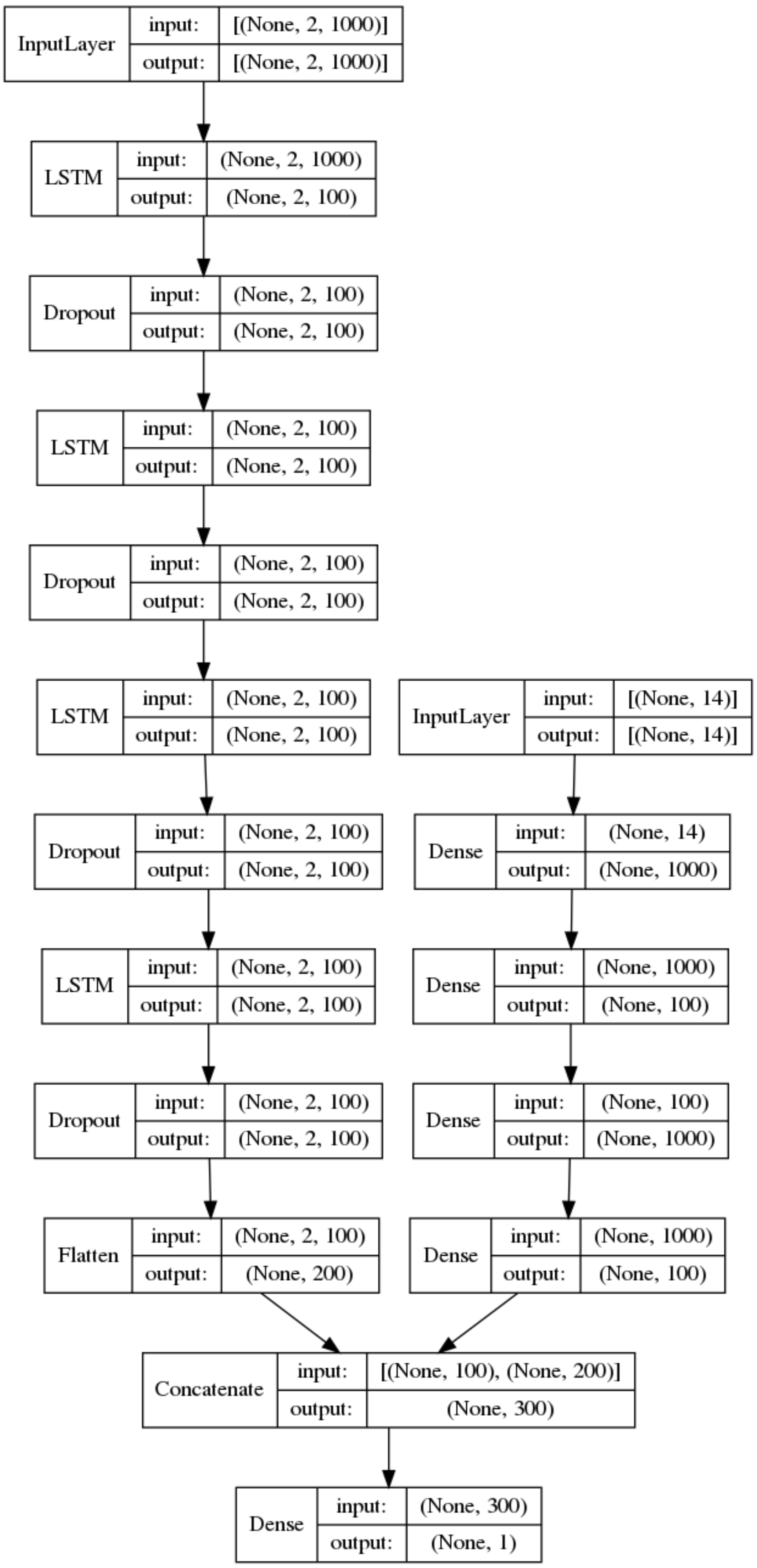
InputLayer | input: [(None, 2, 1000)] | output: [(None, 2, 1000)]
LSTM | input: (None, 2, 1000) | output: (None, 2, 100)
Dropout | input: (None, 2, 100) | output: (None, 2, 100)
LSTM | input: (None, 2, 100) | output: (None, 2, 100)
Dropout | input: (None, 2, 100) | output: (None, 2, 100)
LSTM | input: (None, 2, 100) | output: (None, 2, 100)
InputLayer | input: [(None, 14)] | output: [(None, 14)]
Dropout | input: (None, 2, 100) | output: (None, 2, 100)
Dense | input: (None, 14) | output: (None, 1000)
LSTM | input: (None, 2, 100) | output: (None, 2, 100)
Dense | input: (None, 1000) | output: (None, 100)
Dropout | input: (None, 2, 100) | output: (None, 2, 100)
Dense | input: (None, 100) | output: (None, 1000)
Flatten | input: (None, 2, 100) | output: (None, 200)
Dense | input: (None, 1000) | output: (None, 100)
Concatenate | input: [(None, 100), (None, 200)] | output: (None, 300)
Dense | input: (None, 300) | output: (None, 1)

FIGURE A.5: Architecture of RNN-ANN combined neural network

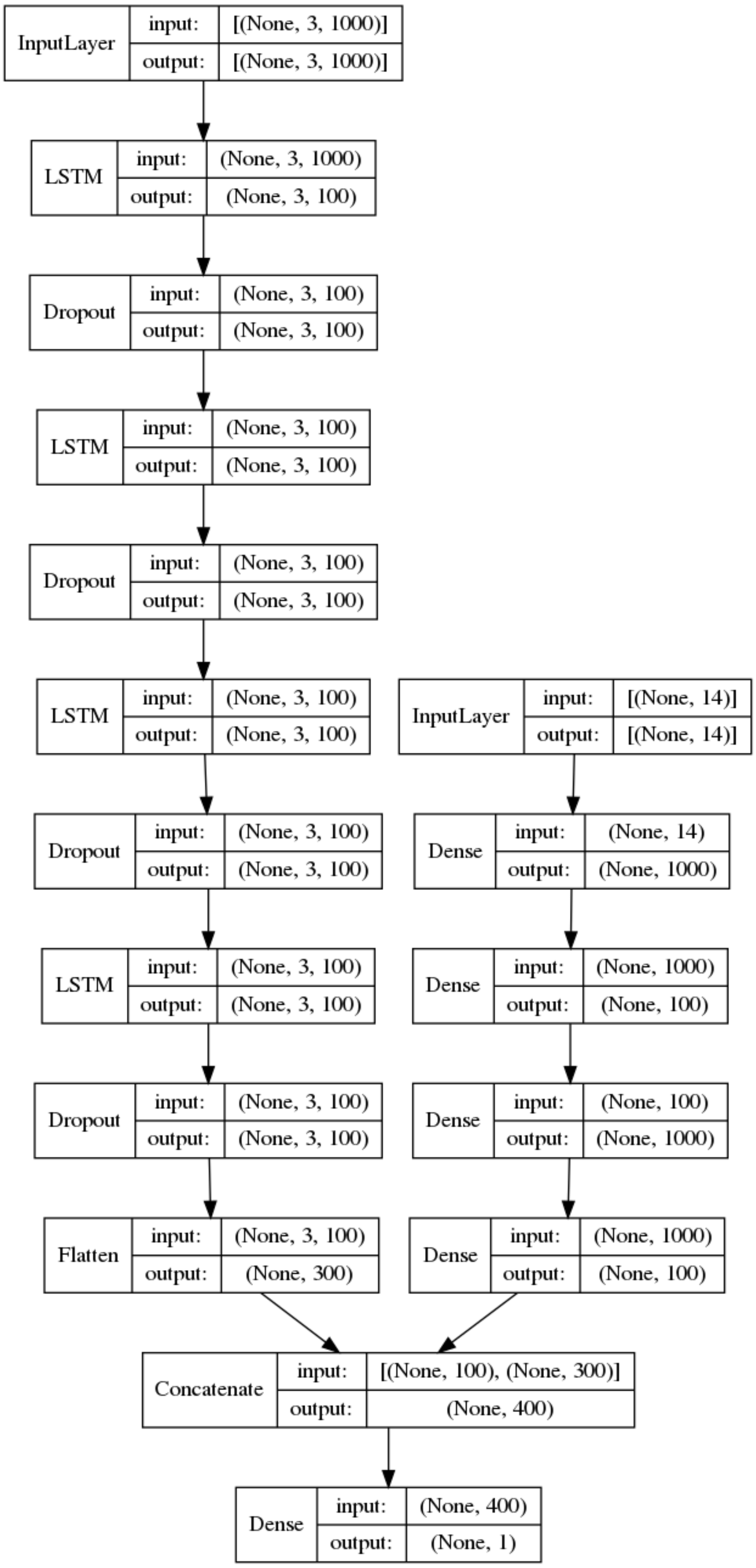

FIGURE A.6: Architecture of enhanced RNN-ANN combined neural network

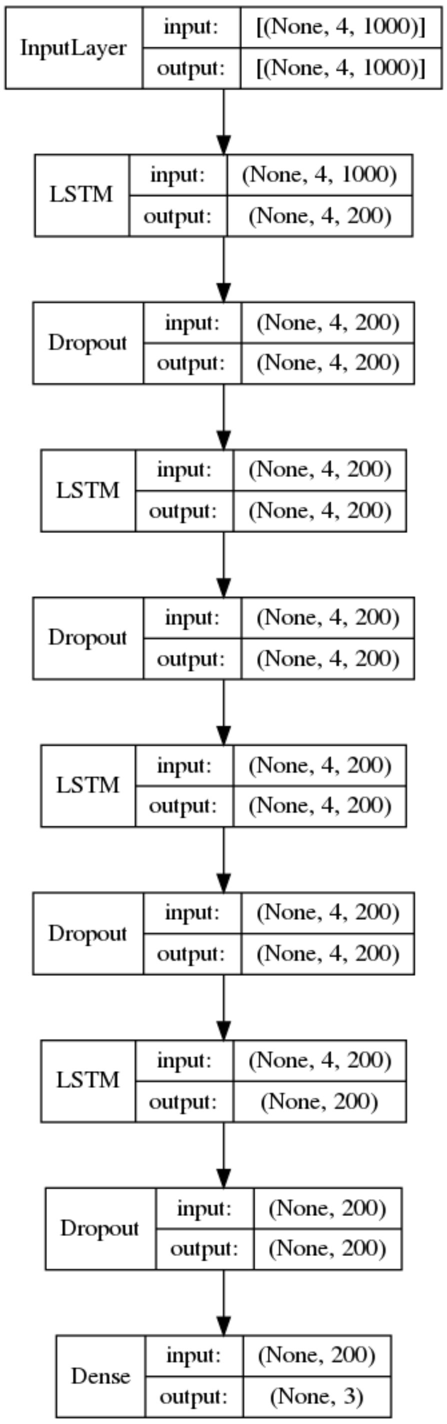

FIGURE A.7: Experiment 2 RNN architecture for ancgle change over time prediction

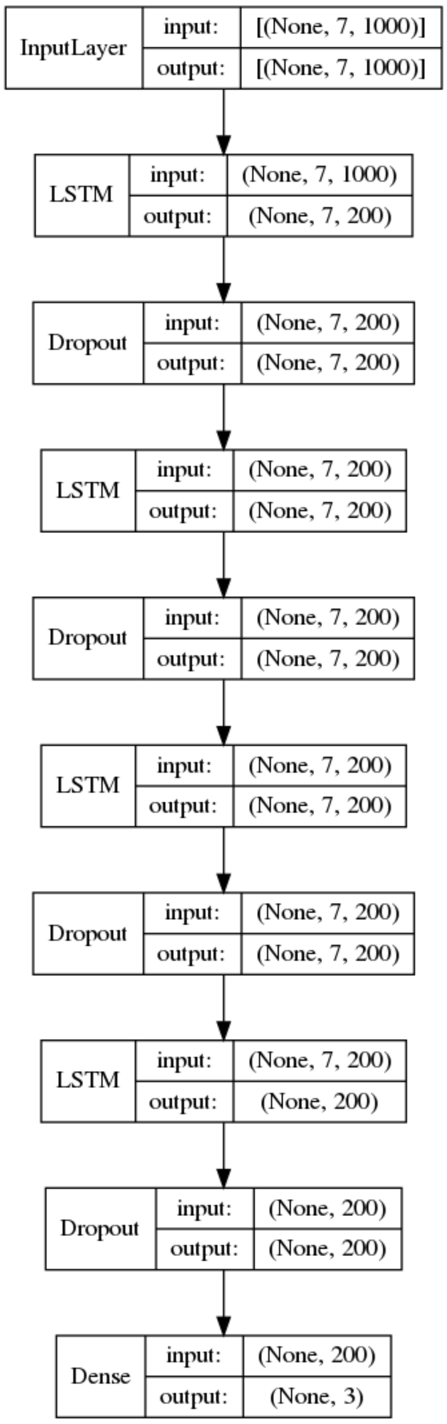

FIGURE A.8: Experiment 2 Enhanced RNN architecture for angle predictino

Bibliography

[1] Riad Akhundov et al. "Development of a deep neural network for automated electromyographic pattern classification". In: *Journal of Experimental Biology* 222.5 (Mar. 2019). DOI: 10.1242/jeb.198101. URL: https://doi.org/10.1242/jeb.198101.

[2] Valentin Bazarevsky et al. *BlazePose: On-device Real-time Body Pose tracking*. 2020. eprint: arXiv:2006.10204.

[3] Benjamin C.Fortune. *Low-cost active electromyography*. HardwareX. 2019. URL: https://www.sciencedirect.com/science/article/pii/S2468067219300501.

[4] Don B. Chaffin, Myun Lee, and Andris Freivalds. "Muscle strength assessment from emg analysis". English (US). In: *Medicine and Science in Sports and Exercise* 12.3 (1980), pp. 205–211. ISSN: 0195-9131. DOI: 10.1249/00005768-198023000-00014.

[5] Calvin W. Y. Chan, Sivan Almosnino, and Evelyn L. Morin. "Wavelet frequency-temporal relative phase pattern analysis for intermuscular synchronization of dynamic surface EMG signals". In: *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2011, pp. 5032–5035. DOI: 10.1109/IEMBS.2011.6091220.

[6] Ulysse Côté-Allard et al. *Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning*. 2018. eprint: arXiv:1801.07756.

[7] M Hallett, B T Shahani, and R R Young. "EMG analysis of stereotyped voluntary movements in man." In: *Journal of Neurology, Neurosurgery & Psychiatry* 38.12 (1975), pp. 1154–1162. ISSN: 0022-3050. DOI: 10.1136/jnnp.38.12.1154. eprint: https://jnnp.bmj.com/content/38/12/1154.full.pdf. URL: https://jnnp.bmj.com/content/38/12/1154.

[8] István Ketykó, Ferenc Kovács, and Krisztián Zsolt Varga. "Domain Adaptation for sEMG-based Gesture Recognition with Recurrent Neural Networks". In: (2019). DOI: 10.1109/IJCNN.2019.8852018. eprint: arXiv:1901.06958.

[9] M.S. Hussain M.B.I. Raez and F. Mohd-Yasin. "Techniques of EMG signal analysis: detection, processing, classification and applications". In: Biological Procedures Online, 2006. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1455479/.

[10] Whited L. Mitchell B. *Anatomy, Shoulder and Upper Limb, Forearm Muscles.* StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing, 2021. URL: https://www.ncbi.nlm.nih.gov/books/NBK536975/.

[11] Christian Morbidoni et al. "A Deep Learning Approach to EMG-Based Classification of Gait Phases during Level Ground Walking". In: *Electronics* 8.8 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8080894. URL: https://www.mdpi.com/2079-9292/8/8/894.

[12] I. Nakajima. “Fast Fourier transform analysis of the masseter muscle EMG during reaction to a warning signal”. In: *Electromyogr Clin Neurophysiol* 35.5 (1995), pp. 281–284.

[13] *Pilates Anatomy*. [Internet] Doctorlib, 2019. URL: https://doctorlib.info/yoga/pilates/4.html.

[14] Christopher Spiewak. “A Comprehensive Study on EMG Feature Extraction and Classifiers”. In: *Open Access Journal of Biomedical Engineering and Biosciences* 1.1 (Feb. 2018). DOI: 10.32474/oajbeb.2018.01.000104. URL: https://doi.org/10.32474/oajbeb.2018.01.000104.

[15] *triceps brachii*. The Free Dictionary [Internet]. Medical Dictionary, Farlex and Partners, 2009. URL: https://medical-dictionary.thefreedictionary.com/triceps+brachii.

[16] Ömer Türk et al. “Classification of mental task EEG records using Hjorth parameters”. In: *2017 25th Signal Processing and Communications Applications Conference (SIU)*. 2017, pp. 1–4. DOI: 10.1109/SIU.2017.7960608.

[17] Muhammad Zahak. “Signal Acquisition Using Surface EMG and Circuit Design Considerations for Robotic Prosthesis”. In: *Computational Intelligence in Electromyography Analysis - A Perspective on Current Applications and Future Challenges*. InTech, Oct. 2012. DOI: 10.5772/52556. URL: https://doi.org/10.5772/52556.

[18] Andrea Maria Zanchettin et al. “Kinematic motion analysis of the human arm during a manipulation task”. In: vol. 2. Jan. 2010, pp. 1–6. ISBN: 978-3-8007-3273-9.

[19] Xiaolong Zhai et al. “Self-Recalibrating Surface EMG Pattern Recognition for Neuroprosthesis Control Based on Convolutional Neural Network”. In: *Frontiers in Neuroscience* 11 (July 2017). DOI: 10.3389/fnins.2017.00379. URL: https://doi.org/10.3389/fnins.2017.00379.

[20] Zhang et al. “Real-Time Surface EMG Pattern Recognition for Hand Gestures Based on an Artificial Neural Network”. In: *Sensors* 19.14 (July 2019), p. 3170. DOI: 10.3390/s19143170. URL: https://doi.org/10.3390/s19143170.