Characterization of the Hand Grasping Using Sensor Fusion

Endre Yllö

2022

Master´s Thesis in
Biomedical Engineering

Supervisor: Nebojsa Malesevic

LUND
UNIVERSITY

Faculty of Engineering LTH
Department of Biomedical Engineering

Abstract

This master‘s thesis explores the possibility of constructing a device capable of classifying human arm movements while performing a set of common tasks. Intended for use on patients undergoing rehabilitation, the device collects data using multiple sensors and applies a convolutional neural network in order to detect different activities of daily living. The sensors used by the device are limited to two inertial measurement units and one radar module. Beyond constructing the device this thesis also attempted to gather its own data set for training and evaluate the systems performance. Finally possible improvements and modifications to the system are proposed.

Contents

Glossary

IMU . . . . . Inertial Measurement Unit

RCS . . . . . Radar Cross Section

TP . . . . . True Positive

TN . . . . . True Negative

FP . . . . . False Positive

FN . . . . . False Negative

ANN . . . . . Artificial Neural Network

NN . . . . . Neural Network

CNN . . . . . Convolutional Neural Network

EM . . . . . Electro Magnetic

1 Introduction

Stroke is a large public health concern in the modern world, in the year 2020 Sweden saw about 27000 cases [1]. Recovery guidelines show that about 75% make a recovery but suffer some impairment as a result. Survivors can be left with several kinds of afflictions depending on what region of the brain experienced the stroke. If the left cerebrum was damaged, there might be impairment regarding speech and understanding words. If the right side of the cerebrum was damaged the patient may experience changes in vision. Another common consequence of stroke originating in the cerebrum is motor function impairment or paralysis, often affecting the opposing side of the body to where the stroke originated in the brain [2].

In order to aid the recovery and increase quality of life in patients suffering from this kind of motor impairment, rehabilitation is vital. However it requires extensive use of the affected limb to regain functionality, this is often a problem due to patients' tendency to rely more heavily on their healthy limb to perform actions. In order to aid and assess rehabilitation it is important to gather qualitative and quantitative information on how the limb is used in day to day life. This will allow for analysis of what actions the patient has done and how they were executed, providing insight into the rehabilitation process. Therefore a wearable device which collects information on how the paretic limb is used and what actions it performs could prove to be of great value.

This project aimed to construct a device to be worn on the arm, capable of gathering information on its movements. In addition, a computational method was constructed capable of accurately discerning what type of action was performed. The device utilized two inertial measurement units (IMU) and a radar module for gathering information combined with a convolutional neural network (CNN) tasked with classifying the information into different activities of daily living. The project was constrained to classify 19 common actions performed in everyday life, using data gathered by the device on test subjects.

Human activity recognition is a field of study concerned with identifying movement patterns by using sensor data. It has seen vast improvement enabled by the availability of sensors and large developments in artificial neural networks. Often taking the form of multivariate time series classification, enabled by artificial neural networks [3][4]. Human activity recognition can find application in rehabilitation where the activity of patients is essential when prescribing optimal therapy. Part of the challenge when working with human activity recognition is the large variation in how movements are performed by different individuals [5].

By allowing this wearable device to collect information and feeding it to the neural network it is the goal of this thesis to explore the possibility of usage in rehabilitation. This is done by collecting data from several people performing actions while being recorded in order to gain data and using it in a convolutional neural network to analyze performance.

2 Background

2.1 Upper limb functionality

In order to better understand the loss of functionality in the upper limb due to stroke, it is useful to understand how a healthy limb functions. The human arm is capable of limited rotation along all three axes and it is useful to understand how positions and readings of sensors corresponds to arm movements. The position of an arm can be described by the positions of the major bones contained within, these bones are pictured in figure 1.

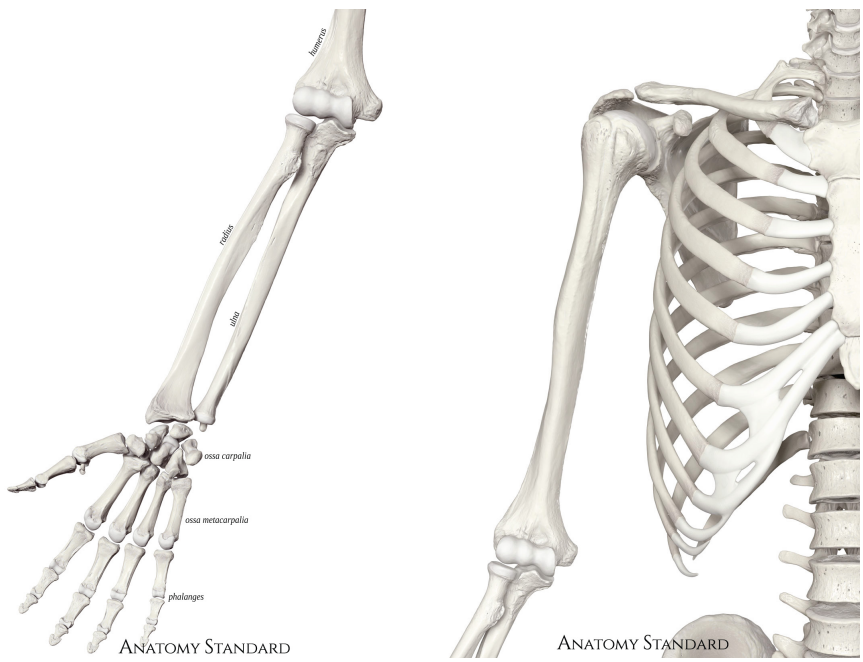

Figure 1: Bones of the arm, used on the basis of creative commons [6].

The upper arm is capable of moving in a wide range of motions centered around its connection to the shoulder via the glenohumeral joint. It is not only capable of movement around this anchor point but also of rotation around the humerus longitudinal axis during external and internal rotation of the shoulder.

For the limited scope of this paper the movements of the lower arm can be broken down into three core movements. First, the flexion and extension via the elbow joint. Second, the rotation of the wrist, supinating and pronating the forearm. Lastly, the flexion and extension of the wrist. All the above mentioned movements are important when assessing how the upper limb is operating and should therefore be taken into account when gathering information.

Differences in people's arms is also a relevant factor to consider. Especially in the aspect of sensor placement, it is necessary to place sensors in similar positions across subjects.

2.2 Inertial measurement unit

IMUs are used to measure the kinematics of an object it is attached to. Each unit consists of three-axial accelerometers and three-axial gyroscopes. There are additional three-axial magnetometers, however they were not used in this project.

The accelerometers measure linear acceleration, comprised of both the sensors movement and the effect of gravity. Gyroscopes measure angular velocity, which is comprised of the sensors speed of rotation. The IMUs used for this project are BNO080 by Bosch (Gerlingen, Germany) mounted on accompanying SparkFun VR IMU Breakout board by Sparkfun (Boulder, Colorado, USA) pictured in figure 2.

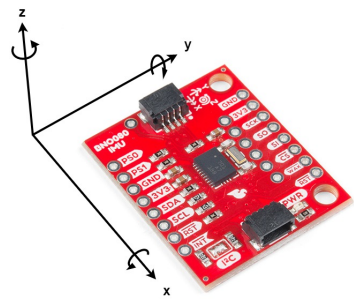

Figure 2: IMU used for the project, used on the basis of creative commons [7].

2.3 Radar

By emitting radio waves and then recording the waves after they have been reflected by the environment it is possible to gather various information about the environment, such as distances, velocities and compositions of objects reflecting the waves. A system that acquires information in this way is referred to as a radar, which stands for radio detection and ranging.

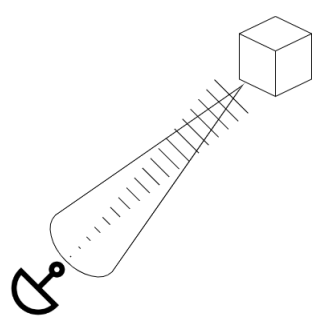

Figure 3: Illustration of radar concept.

A radar system can be used and implemented in various ways, this project utilized a radar system to detect and determine the distances of objects from the sensor. The foremost object to be detected in this implementation is the dorsal side of the hand during wrist extension.

2.3.1 Radar cross section

In order for a radar system to detect an object it is of great importance that the object in question reflects a significant amount of the transmitted radar pulse. This reflective property of an object is referred to as its radar cross section (RCS). An object's RCS mainly depends on three different factors, interception, reflection and directivity [8].

Antennas radiate electromagnetic (EM) waves in specific patterns based on the structural properties of the antenna. This pattern of radiation is described by lobes of varying strength, but is often dominated by an intended main lobe of radiation aimed towards the object to be detected. Interception of a radio wave depends on how much of the transmitter's main lobe intercepts the target object.

Once an object is intercepted by incoming EM waves only a portion is reflected back in the same direction as the source. Some of the waves will be absorbed or simply propagate through the object without interacting. To what extent EM waves are reflected depend on the material properties of the object. If the object is made out of a conducting material, more of the EM wave will be reflected than if the object is made out of a material possessing dielectric properties. The surface of the object is also of importance in how EM waves reflect off it. Smoother surfaces will more uniformly direct the reflection in what is called specular scattering, rougher surfaces will reflect EM waves in several directions called diffuse scattering. The difference of these scattering types is shown in figure 4 [8].

Most objects do not look the same from every angle, unless they are spherical. Which means that the orientation of the object facing the radar will likely impact its RCS.

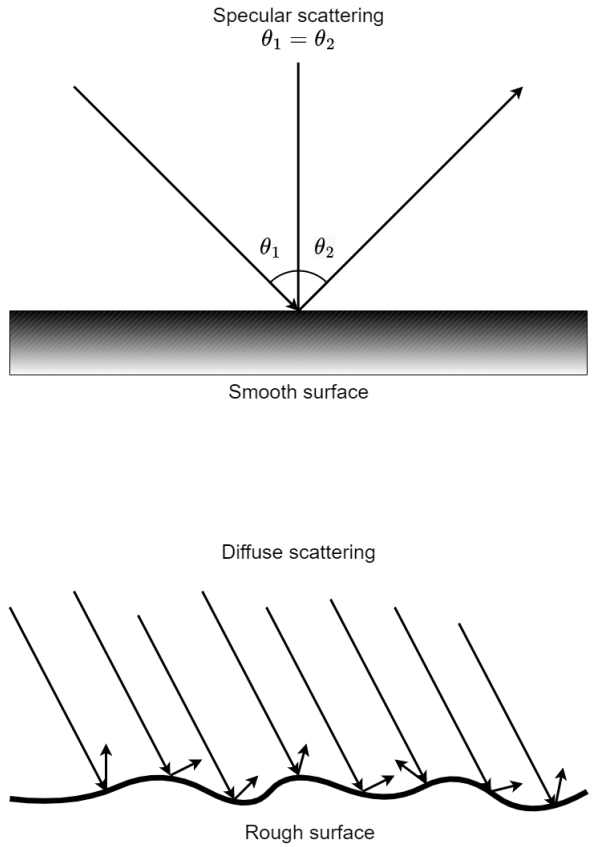

Figure 4: Illustration of specular vs diffuse scattering.

2.3.2 Radar envelope curve

In order to calculate the distance to an object the radar measures the time it takes for the EM wave to propagate towards an object and then propagate back to the radar system. Knowing the speed at which the EM wave travels, which is close to the speed of light in vacuum, enables the radar to calculate the distance to the object. In practice there are often several objects within the radar field of view, placed at different distances, possessing differing RCS. When the radar emits an EM wave and then proceeds to listen for reflections it will receive EM waves of varying delay, representing distances, and magnitude representing RCS. By plotting the strength of returning EM signals and plotting them over time it is possible to construct a radar envelope seen in figure 5.

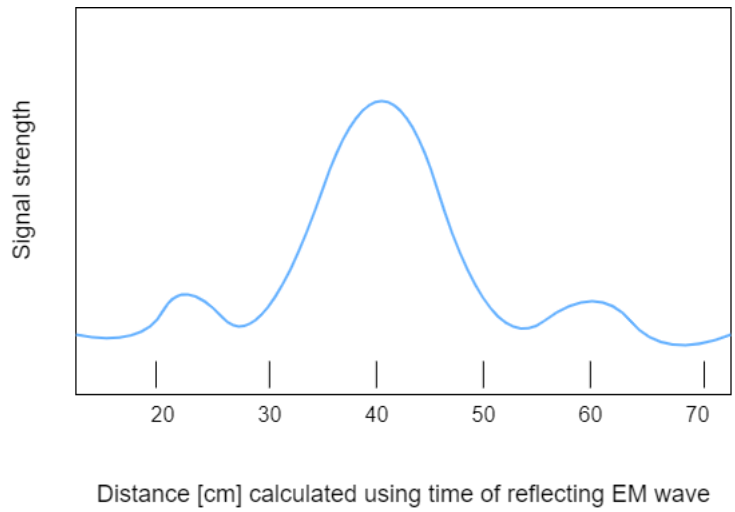

Figure 5: Example of radar envelope.

2.3.3 Acconeer radar module

The XM112 radar module from Acconeer (Lund, Sweden) is built around the A111 60 GHz pulsed coherent radar, it features high precision distance measurements and update frequency. A pulsed radar utilizes the advantages of emitting a pulse of EM wave as opposed to a continuous emission, granting the radar a lower power consumption. A coherent radar has a stable time and phase reference allowing for higher accuracy in measurements. The pulsed coherent radar combines these two methods in order to gain the best aspects of both approaches. More information about the radar module can be found in the modules accompanying data sheet [9] [10].

2.4 Machine learning

Machine learning aims to use data in order to have a computer program recognize different patterns within data without using explicit instructions. This is achieved by having a computer program automatically adjust itself based on observations and examples called *training data*. Machine learning is typically divided into the three categories, supervised learning, unsupervised learning and reinforced learning. Supervised learning relies on labeled data and attempts to assign data to known labels. Unsupervised learning does not use labeled data, instead it infers it´s own self organized labels. Reinforced learning focuses on interaction with an environment and uses a reward mechanism as feedback in order to make beneficial decisions. This project used a supervised machine learning model due to the ease of labeling while recording clearly defined actions.

2.4.1 Supervised machine learning

Supervised machine learning utilizes training data that contains examples of how specific input results in specific output, and the goal function of the machine learning system after training is to map known input variables $\mathbf{x}$ to unknown output variables $\mathbf{y}$.

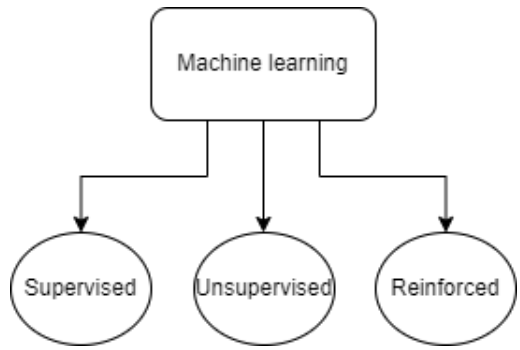

Figure 6: Three sub categories of machine learning.

In order for the machine to be able to construct a function capable of mapping **x** to **y** it requires a large amount of training data providing examples. This demand for training data is increased when the set of possible **y** mappings is large or if the statistical properties differentiating them are too complex to easily separate based on inputs **x**.

The output **y** is referred to as a label and represents a higher fact or quantity associated with input data points. Labels can take on many forms such as numerical values or more abstract concepts such as the state of what is being observed. As an example, this project will take inputs relating to how the arm moves and use this data to label movements with actions such as answering a phone call.

2.4.2 Classification

A common task for a machine learning model is to classify data points into categories. Machine learning models dedicated to this task are often focused on classifying objects in pictures such as cars or animals. The complexity of solving this type of problem will increase with the number of classes and how similar they are. As an example it is a less complex problem to correctly classify pictures displaying the two categories of boats and cars, but it is considerably harder to categorize pictures of several different car models. Partly because of the high similarities of cars and partly because of the higher number of car models.

In order to evaluate the performance of a system tasked with classifying data it is useful to implement a confusion matrix. The confusion matrix is a way of displaying how the machine learning model performed on test data. A confusion matrix is constructed by logging what the model guessed the class to be versus the actual class the data points belong to in a matrix. This aids in discovering classes that are commonly mistaken for each other, there are also several performance metrics that can be calculated from the data in the confusion matrix. These metrics are calculated using the rate of the four indexes, *True positive* (TP), *True negative* (TN), *False positive* (FP) and *False negative* (FN). These indices are organized in a manner seen in figure 7. Take note that the cell labels are dependent on which class is being evaluated, however they follow the geometric pattern outlined by the color coding in figure 7 which is labeled for the evaluation of class *C*.

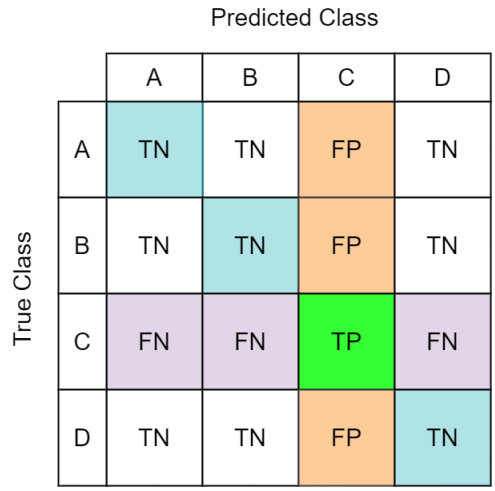

Figure 7: Confusion matrix example labeled for evaluation of class *C*.

There are several performance metrics that can be calculated using the above mentioned indexes, this report will mainly focus on the following.

- *Accuracy*: The proportion of correct guesses positive or negative made by the model in relation to the total amount of guesses made. Calculated once for the whole matrix, this performance metric calculates the proportion of guesses that are placed in the top left to bottom right diagonal of the confusion matrix. Pictured in figure 7 with blue and green. This metric is an evaluation of the matrix as a whole and will therefore not be calculated with any single class in mind.

- *Precision*: The proportion of correct positive predictions by the model in relation to total positives. That is to say the percentage of correct guesses made out of positive outcomes. This metric represents an aspect of the performance of a single class, and can thus be calculated once for every class contained in the matrix. The calculation is presented in equation 1, where precision is evaluated for a specific class indicated by the subscript *k*. Note that this calculation utilizes information contained in the confusion matrix column of the class being evaluated, marked by an orange and green background in figure 7.

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{1}$$

- *Recall*: Proportion of correct positive guesses in relation to everything that should be predicted as positive in an ideal system. This metric is also class specific and thus calculated for every class *k*. Calculated according to equation 2 it utilizes information gathered from the true instances of a specific class pictured in figure 7 as the purple and green horizontal line of cells.

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{2}$$

- *F1*: The harmonic mean of precision and recall for a specific class. Being a combination of the two previous metrics F1 gives an estimate of the systems performance for a single class that takes both aspects into account. Calculated according to equation 3.

$$F1_k = \frac{2 * Precision_k * Recall_k}{Precision_k + Recall_k} \tag{3}$$

In addition to the three previously mentioned class specific performance metrics their respective averages can be used to evaluate the confusion matrix as a whole. These averages take the prefix *macro* and are calculated as the arithmetic mean of the respective metric according to equation 4 using *F1* as example and where *K* represent the total amount of classes.

$$macroF1 = \frac{\sum_{k=1}^{K} F1_k}{K} \tag{4}$$

When tasking a model with classification it is of importance to ensure that each class is equally represented in the data set. It is of course important that the model has enough examples to learn from, but beyond that the instances of classes in relation to each other comes into play. When one or more classes is over represented in the training data set it can lead to a bias developing in the model, where it in cases of severely skewed data sets can reach a high performance by simply guessing on the most over represented class regardless of what the data points suggest. In order to mitigate this issue, it is good to aim for an equal distribution of classes in the training data set so that no class is significantly over or under represented [11].

2.5 Artificial neural networks

Artificial neural networks (ANN) have gained a lot of attention in recent years and can be introduced as analogous to their natural counterpart, biological neural networks from which they were first conceptualized. A neuron in the mathematical sense is a nonlinear parameterized function, several neurons linked together form a network of neurons also known as an artificial neural network. In practice we use the perceptron model to describe a neuron that takes weighted inputs and applies an activation function to generate an output. In order for the ANN to achieve its intended goal, whatever that might be, it requires the weights being applied to the inputs to be 'correct'. In order to achieve correct weights the network needs to be trained.

2.5.1 Perceptron

The fundamental building block of an ANN is known as a perceptron. These perceptrons when linked together form an ANN. In order to fully comprehend ANNs it is useful to understand the perceptron model that describes the foundation of these systems.

The perceptron model takes inputs denoted $x_1, x_2, .., x_n$ and multiplies each input x_i with a weight w_i unique to each input, these weights are used as the trainable parameters for the system. Calculating the sum of the products from every input x_i and weight w_i paired together with an additional bias b singular value enters into an activation function. The result of this activation function is the output y of the perceptron.

The full perceptron model is pictured in figure 8 and described in equation 5 [12]. Activation function can vary depending on implementation but its purpose is to map the summation onto a nonlinear output. Examples of activation functions are the rectified linear unit (ReLU) defined by equation 6 and sigmoid defined by equation 7.

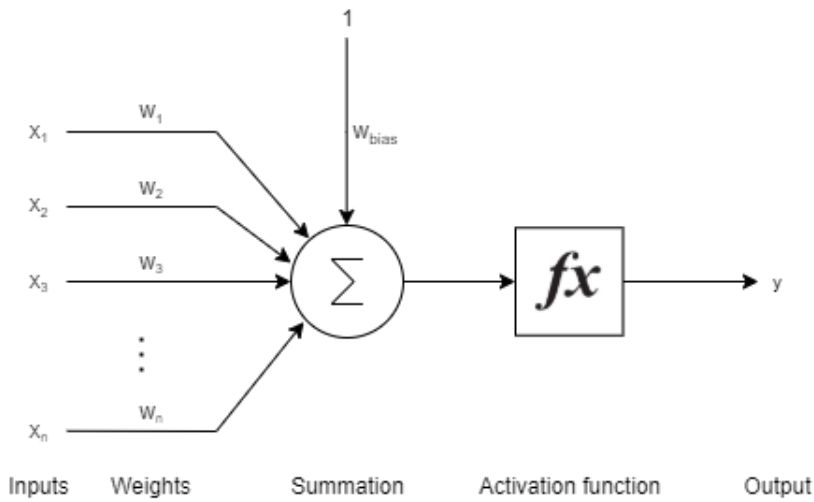

Figure 8: Illustration of the perceptron model.

$$f\left(w_{bias} + \sum_{i=1}^{n} x_i w_i\right) = y \tag{5}$$

$$f(x) = \max(0, x) \tag{6}$$

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

2.5.2 Deep Feed Forward Network

By arranging several perceptrons in a layer and connecting each perceptron output of a layer to the input of every perceptron in the next layer, one can create a several layers deep neural network. The varying sizes of the layers and the total number of layers will correspond to the complexity of the network, more complex networks enable the network to carry out more complicated tasks. An example of this layered architecture is pictured in figure 9, where each circle represents a node that calculates the sum of input arrows and maps it to an activation function. Each arrow represents a pathway and has a weight associated with it.

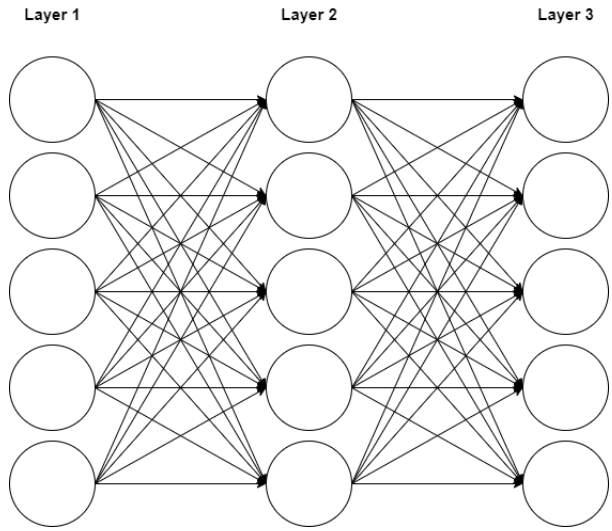

Figure 9: Neural network architecture, consisting of three layer each containing 5 neurons. Configured in a feed forward manner.

Since every layer only propagates outputs 'forward' into new layers and does not feed back into earlier layers it is referred to as a "feed forward network", opposed to a recurrent neural network that can feed some information "backwards" forming a loop.

Once several of these layers are placed sequentially and interconnected in a feed forward manner, they are referred to as hidden layers. This structure handles the bulk of the problem solving ability of the system. In order to reduce computational complexity the hidden layers can be coupled with input stages.

2.5.3 Convolutional Neural Networks

By placing many perceptrons and connecting them in an organized manner, together with some other functions one can construct the architecture of an ANN.

A convolutional neural network is a type of ANN architecture that has been demonstrated to be capable of achieving good results when applied to classification problems similar to the one presented in this thesis. It is primarily known for classifying images, however it has also been successful in the multivariate time series classification type problem presented in this report. Utilizing a CNN architecture comes with an especially useful advantage, namely automatic feature extraction. Features are properties of the data that are used as input to the neural network, examples of features could be the mean value or variance. Extracting them automatically frees the user from extracting useful features in the data and feeding them to the ANN, instead the CNN is capable of extracting features on its own.

A CNN is based around the convolution operation, presented in equation 4. Where x is the input and w is a weight function. This equation is essentially generating a weighted average of the input to generate the output s. From a machine learning perspective, the w is referred to as a kernel which contains the adaptive element, while the output is referred to as a feature map [13].

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{8}$$

The convolutional step is often coupled with a pooling layer. Pooling layers are used to reduce the amount of information while preserving the underlying feature. In max-pooling this is done by selecting the maximum value in a subset of the data and describing all data points in the subset with this number. Other pooling operations may take the average of the subset, such as average pooling [14].

Other relevant layers are the dropout and flattening layers. Dropout layers simply remove certain randomly selected inputs, at a user defined rate. Additionally it scales up the remaining inputs so that the total sum over all inputs remain unchanged. The purpose of a dropout layer is to prevent overfitting. Flattening layers are used to collapse the dimensions of the input into a one dimensional array. This is done in order to satisfy the input requirements of upcoming layers.

2.5.4 Preprocessing

In order for the CNN to perform well it is important to perform preprocessing on the input data in order to increase legibility. An optimal preprocessing scheme will be task specific, however some preprocessing steps are standard practice, such as normalization.

It is common practice when applying a time series to a classifying CNN to parse the data into time windows that represent each time series over a subset of time. Each of these time windows will represent an example of data belonging to a single category. Time windows can be generated with a time "overlap" in an effort to make efficient use of the available data. Figure 10 illustrates the time windowing process [15]. Where L is the window length and s is the step length separating the start of adjacent time windows.

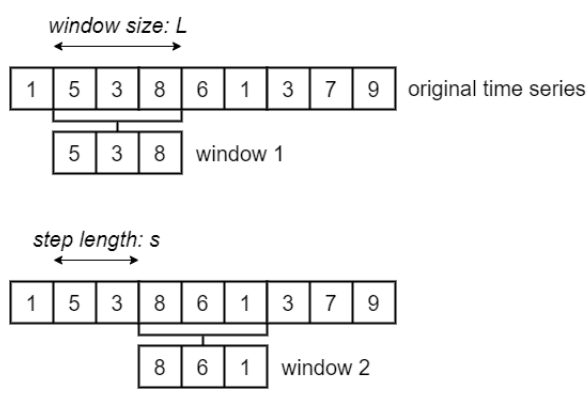

Figure 10: A time series that has two time windows marked with a length of 3 samples, separated by a step length of 2.

Combining several time windows with corresponding timestamps and aligning them into a 3 dimensional matrix creates an input tensor. This tensor will describe the constituent time series values over a specific subset of time. It is wise to select time series that when combined contain useful features in order to aid the ANN. As an example,

this project has constructed 5 tensors separated by sensor module and what information is described.

3 Methodology

This section aims to describe the manner in which the system used to collect data was put together, and how the CNN used for classification was constructed and trained. Firstly the set up of the sensors will be detailed followed by how they come together into the complete data gathering system and lastly the composition and training of the CNN will be described.

3.1 Sensor configuration

The system was centered around a PC running a python script acting as a communication host. The PC hosted two connections, the first leading to a Teensy 4.0 micro controller development board via UART using a baud rate of 500000 *bits/s*. The Teensy was in turn connected to the radar module via another UART connection which operated on a baud rate of 115000 *bits/s*. The second connection to the PC was to a Teensy 4.1 via UART using a baud rate of 115200 *bits/s*. This Teensy communicated with two BNO080 using two SPI channels. Figure 11 illustrates how the system was configured.

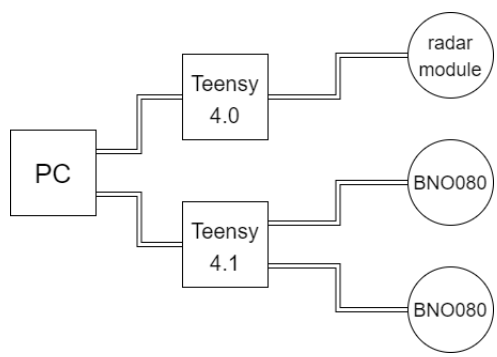

Figure 11: Description of the systems configuration

3.1.1 Implementation of the data acquisition system

The XM112 radar module is capable of performing in several different modes intended to accomplish a variety of tasks such as presence detection or in-phase quadrature. After consideration, the ”power bins” mode was selected. This operating mode was used in order to create an envelope curve. Selecting modes and operating parameters was done by manipulating the modules registry via a UART connection according to the XM112s data sheet. The first objective was to find what parameters would be best suited for the task at hand. It was expected that the primary information sought after by the radar module would be the angle of the wrist. Radar parameters were set until a clear signature of the wrist extension was distinguishable in the envelope curve, while the module was placed at a distance of 10 cm from the wrist. Table 1 below contains the parameters decided upon.

After the parameters used by the radar were decided upon, the next step was to integrate the IMUs by constructing a script logging their output values. No parameters were needed for implementing the IMUs, as they were pre set to stream raw gyroscopic and accelerometer data.

Bins	Start depth	Depth range
1034	100mm	500mm

Table 1: Radar settings.

The script used to log data was written in python utilizing modules *serial* to read sensor data from the UART buffer and *numpy* to organize it. The script also used module *keyboard* to allow for labeling of data during recording and *time* to gain a unified time reference for when each data point was collected. Data was continually saved in .txt files in order to gain a constant sampling rate. The sampling rates for the IMUs and radar are presented in Table 2, these sampling rates were the highest the system could achieve.

Radar	IMUs
18Hz	947Hz

Table 2: Sampling rates.

The last step in constructing the data gathering system was to consecutively store the gathered data in a separate file in order to limit any slowdown of sampling rate during the devices run time due to inefficient storage.

The data from the IMUs were stored in separate files from Radar data, and linked together via an event id shared between them. The IMUs data contains 12 total time series, 6 for each IMU unit. Furthermore it contains a timestamp for each sample and an action log representing what action was performed by the user during every timestamp. The radar data was organized according to an envelope curve, each timestamp has an associated 1034 bins representing radar return from different distances between 0.1 and 0,6m. In addition to this, two unique id signatures were assigned. One for every run of the protocol called ”event-id” and one for distinguishing between users called ”user-id”.

Index	index	event id	user id	activity	timestamp	x axis lower	y axis lower	z axis lower
0	214	100	2	phone call	11.4878	-0.3626…	-0.0064…	-0.4992…
1	215	100	2	phone call	11.5514	-0.3406…	-0.0064…	-0.5662…
2	216	100	2	phone call	11.5991	-0.2713…	-0.0101…	-0.6094…
3	217	100	2	phone call	11.6539	-0.2766…	-0.0138…	-0.5758…

Figure 12: Example of the data structure, containing meta data and acceleration of the lower arm IMU.

3.1.2 Sensor placement

Consideration was made to ensure that sensor placement would be as similar as possible between test subjects. For the radar module, this was done by measuring the distance between the radar and the wrist and placing it with approximately 10cm of space separating them. The IMU intended for use on the upper arm was placed at the

distal region of the upper arm facing inward towards the torso. Placement was below the contour of the bicep as to limit movement caused by its contraction. The second IMU intended for the lower arm was placed in the carpal region below the palm. Further elaboration on sensor placement can be seen in the figure 13. Where the white boxes contain the IMUs and the grey box holds the radar module.

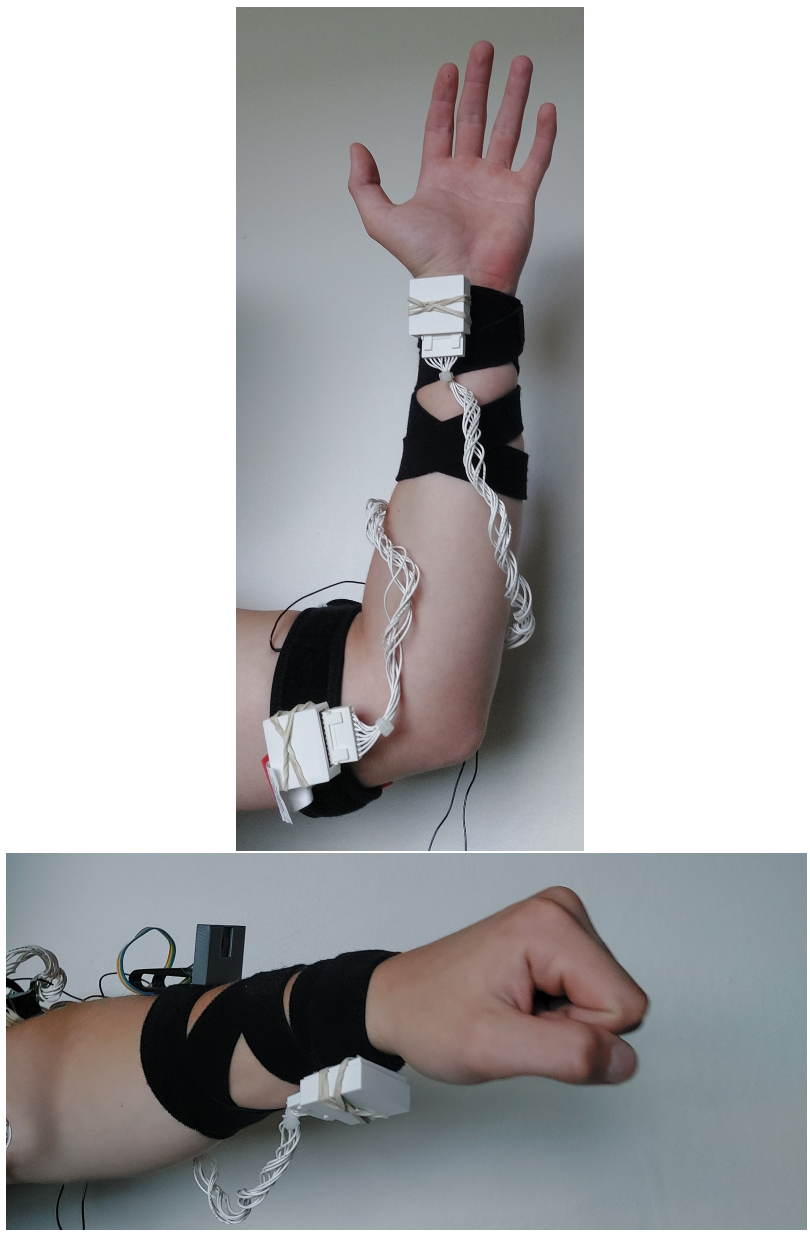

Figure 13: Pictures of sensor placement. The white boxes contain the IMUs and the grey box in the lower figure contains the radar module.

3.2 Protocol

A protocol was made in order to gather concise data to be used during training of the neural network. This protocol consisted of 19 tasks that were perceived to represent everyday actions of a user. A total of 10 people participated as test subjects. Every test subject executed the protocol twice, yielding two data sets per subject each containing a full execution of the protocol. Each task was performed for a duration of 20 *s* and separated by a 10 *s* rest period. More detailed information about the performance of each task is presented in the list below. All tasks were performed sitting down in front of a table, certain tasks involved props in order to increase the realism of the actions. The tasks are mean to represent common movements in everyday life of varying similarity to provide an appropriate challenge for the CNN.

- Phone call: The subject's own phone was picked up from the table and held to the ear by the right arm. The phone was placed back on the table by the end of the action before the resting phase began.
- Texting: The subject's own phone was picked up from the table and used to type any message of the subjects choosing, after which the subject placed the phone back on the table before the next resting phase began.
- Wash hands: The subject held up his/her hands in front of them and pretended to wash their hands in a sink. No actual soap or water used.
- Dry hands: Performed by picking up a towel and drying hands, then placing the towel on the table.
- Wipe counter: performed by taking the towel and cleaning the table, no specific instruction on wiping movement was given.
- Use Tv-remote: Performed by picking up remote and repeatedly changing channels on an imaginary tv.
- Write on paper: The subject was provided with a pen and paper to write on. No instructions were provided on what to write.
- Type on keyboard: The subject typed whatever they deemed fit on a provided keyboard.
- Use mouse: performed by using a provided mouse. No instruction was given on how it was to be used.
- Tying knot: A sting was provided and the subject was instructed to tie any type of knot on it.
- Brush teeth: The subject pretended to brush their teeth, no toothbrush was used.
- Brush face: The subject was instructed to apply imaginary makeup or shaving foam to their cheeks.
- Comb hair: Performed by pretending to comb their hair, no comb was used.
- Open bottle: A bottle was provided and the subject was instructed to repeatedly open and close it.

- Drink water: Using the same bottle while the lid was closed, the subject pretended to drink from it in several swigs.
- Eat with spoon: The subject was provided with a spoon and pretended to eat something with it.
- Take medicine: The subject was instructed to take several imaginary pills placed on the table.
- Button shirt: The subject was instructed to button and unbutton their clothing, if the subject did not wear any garment with buttons the task was mimicked.
- Resting: The subject was instructed to rest their arm on the table and keep it as still as possible.

Detailed instructions to the test subject on how to perform each task was avoided, in order to generate data that better reflects a person's natural behavior.

3.3 Neural Network

Once the training data had been collected, the next objective was to generate a neural network model capable of classifying the tasks with the highest possible accuracy. In order to achieve this, several different network architectures were tested together with different preprocessing methods, to see what would yield the best results.

3.3.1 Preprocessing

A script was made to fix certain errors that occurred during the recording, caused by accidental logging of an action during the execution of the protocol. It was noted that accidental logging of the wrong actions was most common during the first second of starting a new task. The accidental pressing of the wrong key would quickly be remedied by pressing the correct key, however the error would still be present in the data set. The script searches the data sets and marks locations where an action was performed for an unreasonably short amount of time. It achieved this by searching for actions that took place for less then approximately 2 s and noting their position in the database. By examining the locations of expected errors provided by the script and comparing them to what was supposed to be logged according to the protocol. The data set was then pruned and altered on a case by case basis.

In order to feed the data from the separate sensors to the CNN it is vital that each time series is of the same length. In order to achieve this, the time series collected from the IMUs were decimated to a point where the only samples kept in these times series had a corresponding sample from the radar taken at approximately the same time. This was done by matching the timestamps associated with each sample.

In an effort to improve computation time, the data provided by the radar was reduced via summing several bins into one. This reduced the data used to describe the radar envelope by a factor of 20.

The next preprocessing step aimed to reduce the over representation of the resting action. This was done in two steps, firstly the data collected in resting state before the first task and after the last task was removed from the data set. Secondly resting data points were removed from the start and finish of each rest event, keeping a proportionate amount of data in the middle of each resting event in order to properly represent the class. The now proportionate but scattered resting data points were sorted and placed

in a group. This was done in order to support the upcoming windowing operation that would otherwise have suffered from additional partitioning of classes. A simplified illustration of this process is found in figure 14 using only two actions separated by resting.

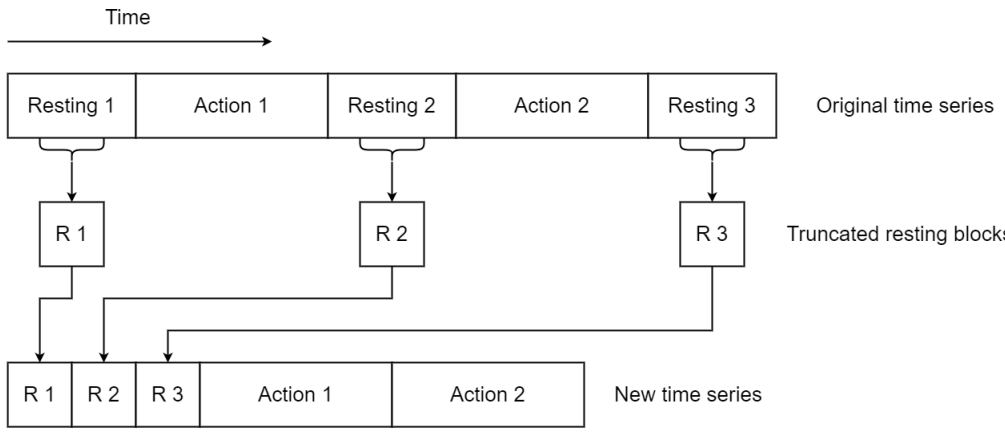

Figure 14: Truncating and organising resting data.

The next preprocessing step was to normalize the data in every time series used as input by scaling every data point in a time series to range between -1 and 1. This was followed by chunking it into time windows. Each time window had a window size of 54 samples and used a step length of five. This corresponds to a window size of 3 s and a step length of 0.28s when measured in time. The time series were grouped into 5 separate tensors, two describing the accelerometer data for each IMU, two more containing gyroscopic data for each IMU and lastly one time series containing the radars envelope curve.

The training and testing data was partitioned in a manner that assigned a person's protocols so that one was represented in the testing set and one in the training set. Resulting in the 50/50 split in training and testing data shown in figure 15. This concludes the preprocessing done prior to feeding the data to the CNN.

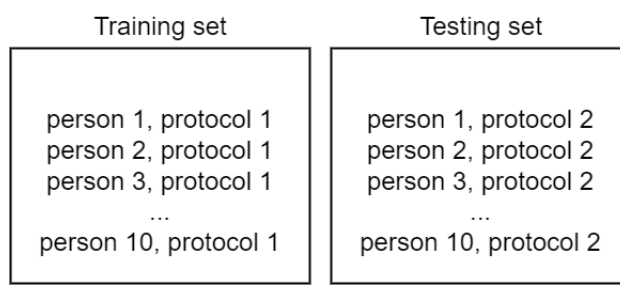

Figure 15: Partitioning of data into training and testing sets.

Other combinations of preprocessing schemes and CNN architectures were tested, in order to see their impact on the results and gain further insight about the systems performance. The alterations tested are listed below.

- Instead of having every person represented in both training and testing data sets, the partitioning was organized so that no person was represented in both data sets. Maintaining the 50/50 split in training and testing data shown in figure 16.

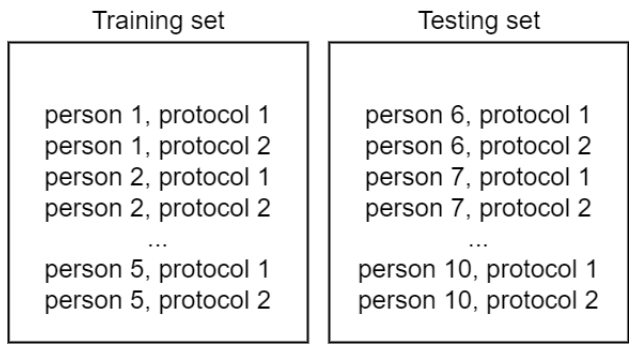

Figure 16: Alternative partitioning of data into training and testing sets.

- Altering the input tensor by removing the radar envelope data and using the IMU data set before decimation, therefore containing higher sampling rate. While using the original train/test separation shown in figure 15. The time windowing process used the same window size and step length in respect to time.
- Altering the input tensor by removing IMU data and only train and test it using radar envelopes. While using the original train/test separation.

3.3.2 CNN

The CNN features a 5 headed design where input tensors are separated based on what is being measured and from what module it originates. This allows for feature extraction to be done on each head separately, it also enables simple modifications to the architecture. Such as omitting specific sensor information in order to evaluate specific parts of the systems performance. After the feature extraction is done, the resulting features are put through max pooling and dropout layers. After this all heads were combined into one tensor and flattened before being fed into the deep neural network layers. The last layer possesses an amount of nodes equal to the amount of classes the CNN attempts to classify. The final layer uses the softmax activation function, as opposed to all previous layers utilizing the relu activation function.

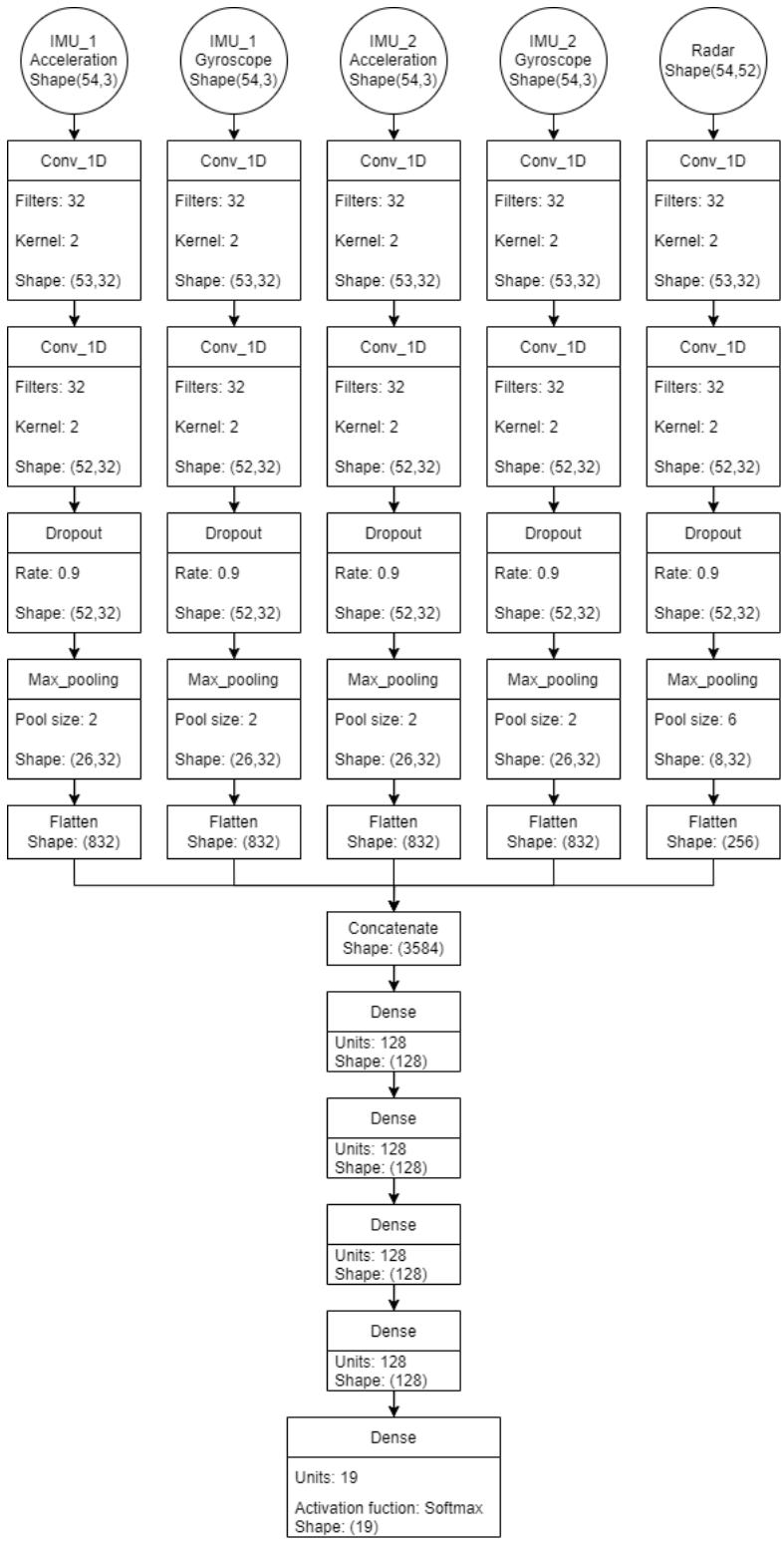

Figure 17: Description of the CNN used to obtain results, activation functions are relu if not specified otherwise. Shape corresponds to the shape of the output tensor.

All data obtained from IMUs were applied to identical feature extraction steps, consisting of 2 convolutional layers defined by 32 nodes each and a kernel size of 2. Feature extraction was continued by applying a max pooling layer of size 2 and finalized by a dropout layer using a rate of 0.9. The data originating from the radar module used a feature extraction identical in structure but with separate parameters. The convolutional layers were consisting of 16 nodes each and using a kernel size of 20. The max pooling layer was of size 6 and the dropout layer used the same rate as the other heads of 0.9. After feature extraction all heads were flattened and concatenated into a single tensor that was fed into the dense layers making up the deep neural network. Each of the layers consisted of 128 nodes and the network was made up of four layers, excluding the final output layer that was made up of 19 nodes. Training of the neural network was done while shuffling the order in which the windows appear.

4 Results

The results will be presented in three parts, the first is going to focus on the performance of the first CNN presented in figures 18 and 19 using decimated IMU data and the radar envelope. The second part will focus on the performance while only using the full IMU data without decimation. The third part will examine the performance of the radar information alone. Each system's performance was evaluated in two separate evaluation methods, once for when each person is represented in both training and testing data and once more for when every person is either represented in training or testing data, not both.

All confusion matrices are normalized over true labels and the values represent percentages. F1 score is calculated over each class. Each result was derived from one training and testing event and accuracies should be compared to the chance level 0.0526.

4.1 Sensor fused data

The first method is presented in figure 18 using decimated IMU data combined with radar data resulting in the following confusion matrices. The first was trained and tested while every person was being represented in both data sets with separate recordings of the protocol. The second result organized the training test separated by individual user id.

This confusion matrix describes performance of the sensor fused CNN. It is followed by a table of performance metrics for every action it attempts to classify.

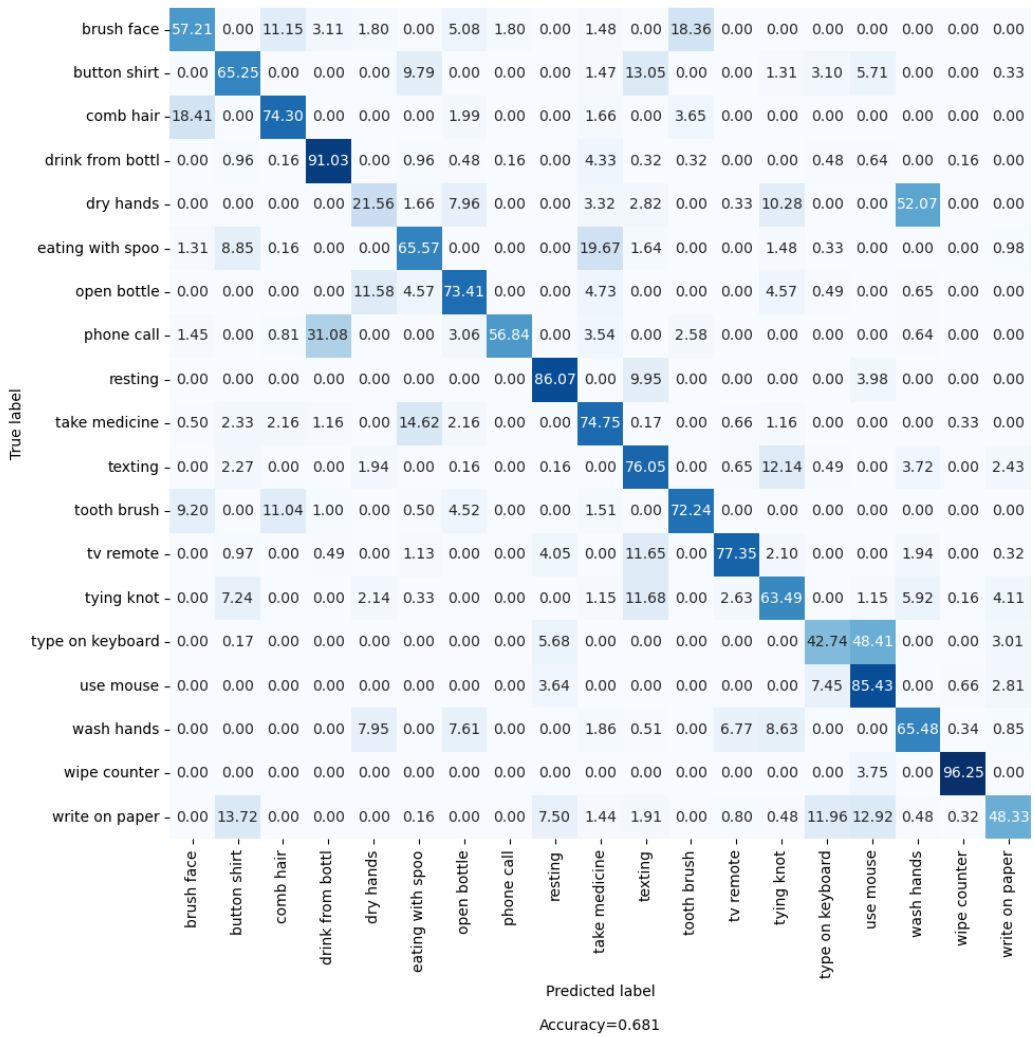

Figure 18: Confusion matrix describing performance of the sensor fused CNN.

Action	Precision	Recall	F1
brush face	0.652336	0.572131	0.609607
button shirt	0.640000	0.652529	0.646204
comb hair	0.744186	0.742952	0.743568
drink from bottle	0.713568	0.910256	0.800000
dry hands	0.457746	0.215589	0.293123
eating with spoon	0.661157	0.655738	0.658436
open bottle	0.693374	0.734095	0.713154
phone call	0.967123	0.568438	0.716024
resting	0.800926	0.860697	0.829736
take medicine	0.614754	0.747508	0.674663
texting	0.588972	0.760518	0.663842
tooth brush	0.739726	0.722408	0.730964
tv remote	0.870674	0.773463	0.819195
tying knot	0.601246	0.634868	0.617600
type on keyboard	0.630542	0.427379	0.509453
use mouse	0.527068	0.854305	0.651927
wash hands	0.494253	0.654822	0.563319
wipe counter	0.979203	0.962521	0.970790
write on paper	0.770992	0.483254	0.594118
Macro Averages	0.691991	0.680709	0.673985

Table 3: Performance of each class for the sensor fused CNN.

The confusion matrix in figure 19 and table 4 describe the performance of the same sensor fused CNN but trained and tested on separate individuals.

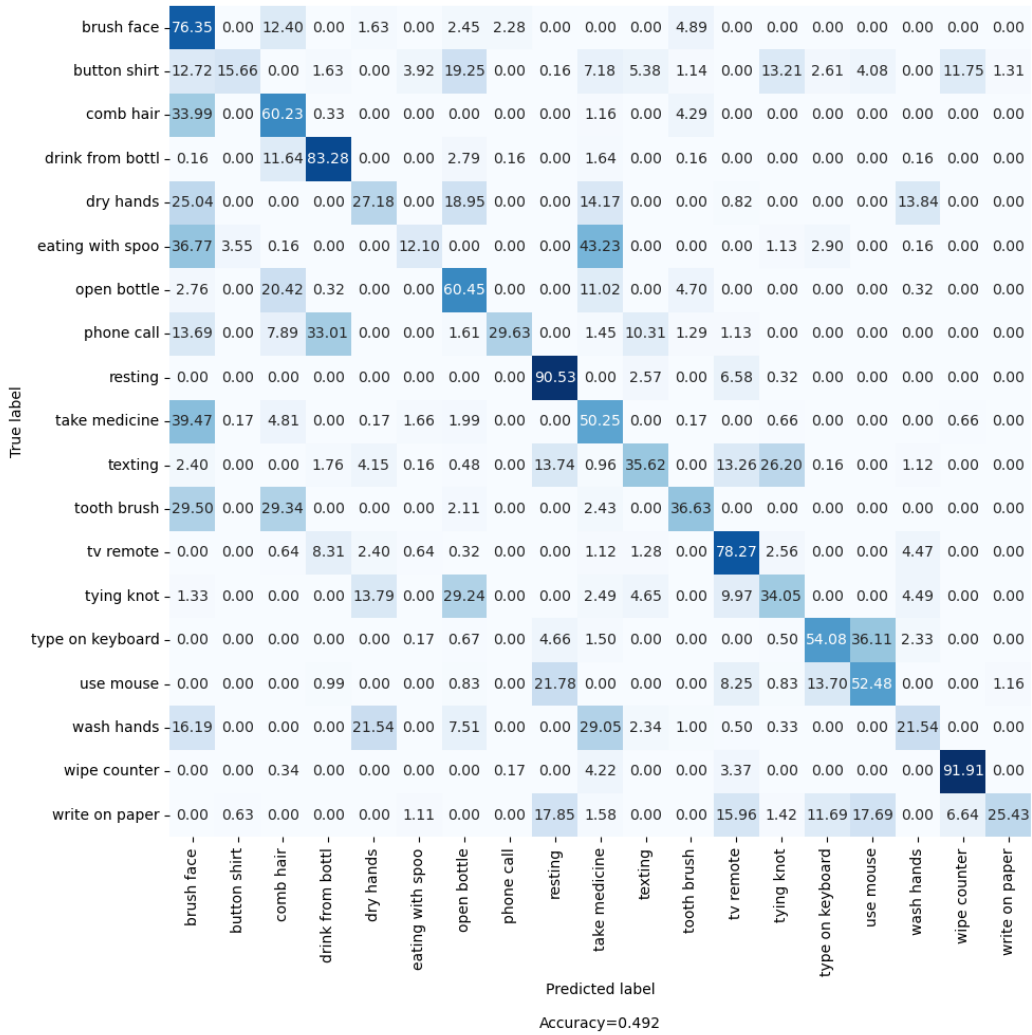

Figure 19: confusion matrix describing performance of the sensor fused CNN with alternative train test separation.

Action	Precision	Recall	F1
brush face	0.263662	0.763458	0.39196
button shirt	0.780488	0.156607	0.26087
comb hair	0.403761	0.602310	0.483444
drink from bottle	0.638191	0.832787	0.722617
dry hands	0.384615	0.271829	0.318533
eating with spoon	0.614754	0.120968	0.202156
open bottle	0.410793	0.604538	0.489180
phone call	0.920000	0.296296	0.448234
resting	0.610390	0.905297	0.729153
take medicine	0.286932	0.502488	0.365280
texting	0.577720	0.356230	0.440711
tooth brush	0.676647	0.366288	0.475289
tv remote	0.569767	0.782748	0.659489
tying knot	0.411647	0.340532	0.372727
type on keyboard	0.628627	0.540765	0.581395
use mouse	0.473214	0.524752	0.497653
wash hands	0.440273	0.215359	0.289238
wipe counter	0.822021	0.919056	0.867834
write on paper	0.914773	0.254344	0.398022
Macro Averages	0.569909	0.492455	0.473357

Table 4: Performance of each class for the sensor fused CNN with alternative train test separation.

4.2 High definition IMU data

This confusion matrix describes the performance of the CNN when excluding radar data and using IMU data before decimation. The CNN was trained and tested while every person was being represented in both data sets with separate recordings of the protocol.

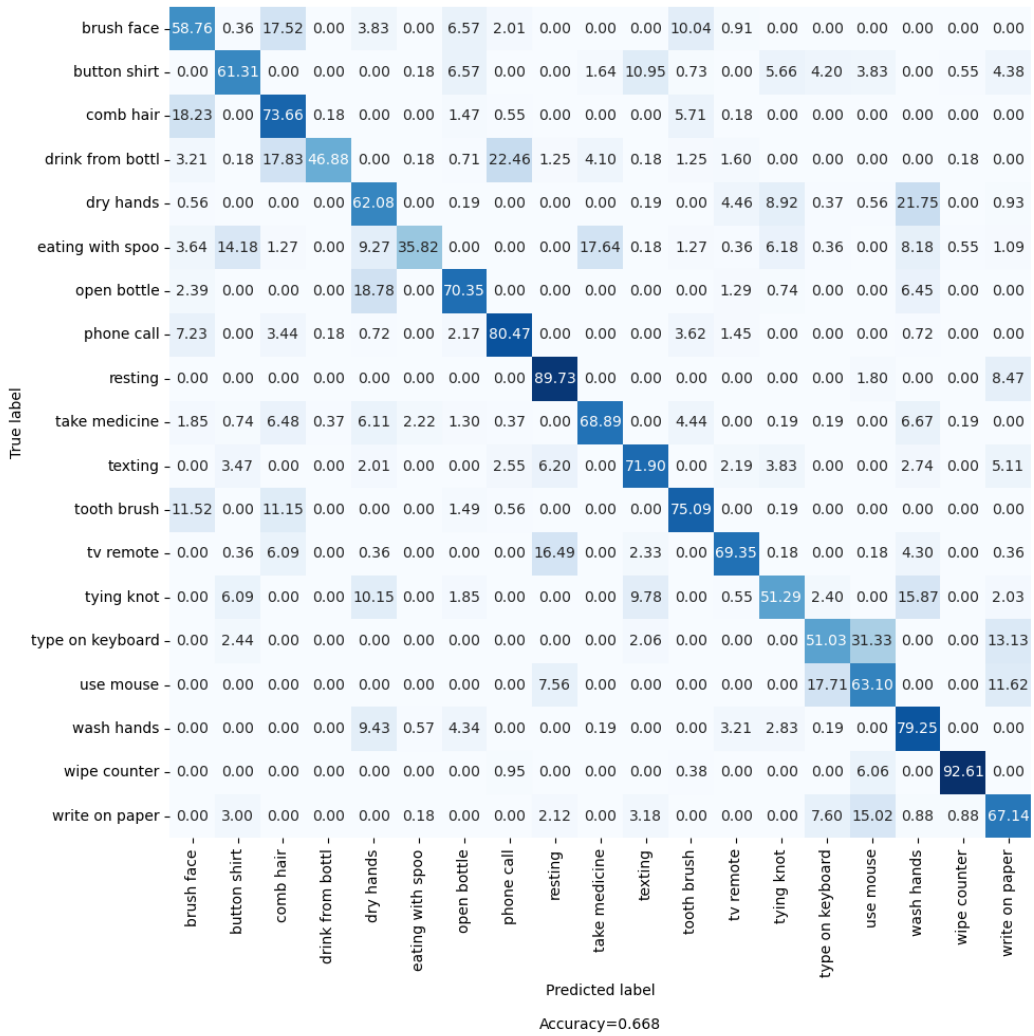

Figure 20: Confusion matrix of performance when using highly defined IMU data.

4.3 Radar Envelope Data

The confusion matrix describing the performance when using radar data exclusively.The CNN was trained and tested while every person was being represented in both data sets with separate recordings of the protocol.

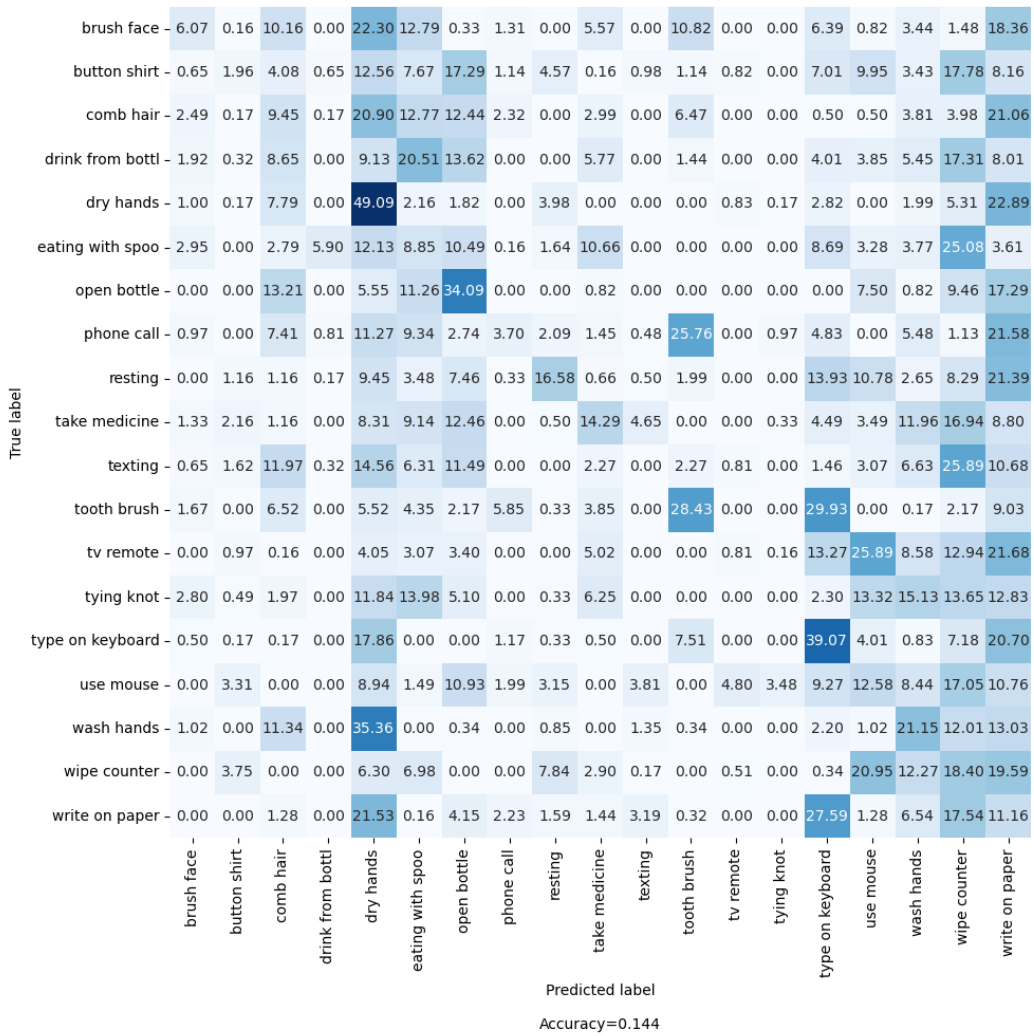

Figure 21: Confusion matrix of performance when using radar data.

5 Discussion

The systems performed well compared to chance level, however it is far from perfect. In the discussion I will further analyze the results and suggest improvements to the system based on the findings.

The result produced by the first CNN architectures seen in figure 18 seems to yield a good result, although not perfect. It shows that the system is indeed capable of classifying many of the actions with reasonable accuracy. When examining the results one can see that errors in classification are more likely for certain actions. One example of this can be seen when the systems attempt to discern between the actions ”washing hands” and ”drying hands”. The fact that these two classes are difficult to tell apart is due to how similar they are, as both of these actions were performed by rubbing one’s hands together while being held at roughly the same angle. It stands to reason that similar actions are more likely to be confused for each other. Other examples of this can be found in the ”type on keyboard” and ”use mouse” actions, in both of these actions the arm is held in very similar positions. Another major reason for misclassifications is likely the limited training data available. This lack of data provided by relatively few subjects is very punishing since subjects vary in how they perform actions. Results would probably improve if each subject performed the protocol many more times. This assumption is supported by the drop in performance when rearranging the test train separation. If the network gets to train on data more similar to the test data it stands to reason that the result will improve. This does also suggest that a lack of training data is at play. If more data produced by different subjects could be obtained for training it is likely that this gap will close somewhat. For real world applications I would argue that the preferential train test data separation is more relevant due to the fact that a unique user’s data can easily be collected and used to train the system further for that person’s way of performing actions.

It is sound to remind oneself of the limited size of the data set before making any major deduction based on the results. My observations during the recording of the protocols were that subjects varied greatly in the manner they executed actions. When one person performs an action in a distinctive manner the risk of the system not being able to learn this association is larger due to the lack of representation in the data set. If the data set would have been larger and recorded on more individuals, the system might have been better able to classify these cases that were previously outliers.

A high variability of results derived from training the same CNN architectures, makes it difficult to present fair results for every architecture. The results presented above were handpicked in an attempt to not present outliers in performance. Certain outlier training’s of the first network were able to achieve accuracies of about 0.8, however these results are not reliably reproducible. This large variance in performances of the same network suggest that implementing a K-fold cross validation method onto the CNNs could provide more relevant results.

The 5-headed CNN design was chosen because it allows for easy modification while keeping the other parts of the architecture consistent. There could be drawbacks when extracting features from so many separate heads if relevant features require a combination of inputs to be recognized.

In order to gain a greater understanding of the system, it is relevant to examine how useful the information gained from the separate sensors are. In order to discern this information I trained similar CNNs while omitting either IMU or radar envelope from the input tensor. These variations aim to see if the system relies more on one type of

sensor data. The result of this can be seen in figures 20 and 21. A poor performance of the method based on radar signals alone can readily be seen. While methods using exclusively IMU data can rival the performance of the full system. This implies that further development should focus on how to make the radar provide more useful information. This can be achieved by altering parameters for the radar envelope by changing its range and resolution. I also suggest changing the operating mode of the radar to other built in options such as presence detection. Another way of improving the quality of the radar information could be to alter the RCS of the hand. This could be done by attaching a material with beneficial properties when it comes to EM reflection. Attaching for example a piece of aluminum foil to the back of the hand would drastically improve RCS providing a more defined feature, however it would likely make the surface more prone to specular scattering. It is also worth mentioning that the radar's performance might be hampered by the protocol, due to certain tasks not involving the relevant tools. The radar would in more life-like applications probably provide some information on what is happening in front of the hand that could prove useful when classifying actions.

Indeed the poor radar performance raises the question if the radar should even be present in the system if it can not be improved, as its current implementation places restrictions on other parts of the system that could prove more beneficial. Two of the restrictions caused by the current implementation of the radar is the sampling rate and complexity of the CNN training. The sampling rate was limited by the radar, so the IMU data was decimated by a factor of 50. This could cause a loss of vital information present in the original IMU data. An alternative option to decimation the IMU data would be to interpolate the radar envelope, however this would severely raise the computational complexity of training the CNN. In order to explore this further we can look at figure 20 presenting the performance of a CNN trained exclusively on IMU data before it was decimated, providing more detailed information to the system that might contain features otherwise lost during decimation. The system could be further improved by using more sophisticated preprocessing methods such as "window warping" as presented in [14]. Where the sliding window is altered to allow for windows to vary in the amount of time they represent, thereby allowing the CNN to explore features that might have been overlooked when using a constant time scale.

References

[1] Socialstyrelsen, https://sdb.socialstyrelsen.se/if_stroke/val.aspx, Accessed 2022-10-17.

[2] Christopher & Dana Reeve Foundation, Stroke (Cerebral Vascular Accident (CVA) and Spinal Stroke), https://www.christopherreeve.org/living-with-paralysis/health/causes-of-paralysis/stroke, Accessed 2022-10-17.

[3] Jiang Y, Song L, Zhang J, Song Y, Yan M. Multi-Category Gesture Recognition Modeling Based on sEMG and IMU Signals. Sensors (14248220). 2022;22(15):5855-N.PAG. doi:10.3390/s22155855

[4] Zia ur Rehman M, Waris A, Gilani SO, Jochumsen M, Niazi IK, Jamil M, Farina D, Kamavuako EN. Multiday EMG-Based Classification of Hand Motions with Deep Learning Techniques. Sensors. 2018; 18(8):2497. https://doi.org/10.3390/s18082497

[5] Bances, E., Karol, A.M.A., Schneider, U. (2022). LSTM and CNN Based IMU Sensor Fusion Approach for Human Pose Identification in Manual Handling Activities. In: Moreno, J.C., Masood, J., Schneider, U., Maufroy, C., Pons, J.L. (eds) Wearable Robotics: Challenges and Trends. WeRob 2020. Biosystems Biorobotics, vol 27. Springer, Cham. https://doi-org.ludwig.lub.lu.se/10.1007/978-3-030-69547-774

[6] Anatomystandard, www.anatomystandard.com, https://creativecommons.org/licenses/by-nc/4.0/, Accessed 2022-09-30.

[7] SparkFun VR IMU Breakout - BNO080, https://www.sparkfun.com/products/14686, https://creativecommons.org/licenses/by/2.0/, Accessed 2022-09-30.

[8] Richards MA, Scheer J, Holm WA, Principles of Modern Radar, SciTech, pp 11-18, 2010.

[9] Acconeer AB, Radar sensor introduction, https://acconeer-python-exploration.readthedocs.io/en/latest/sensor_introduction.html#radar-sensor-introduction, accessed 2022-05-22.

[10] Acconeer AB, XM112 – Pulsed Coherent Radar (PCR) Module Datasheet v1.0, 2019-01-29 , https://developer.acconeer.com/download/xm112-datasheet-pdf/.

[11] Sebastian Raschka, Vahid Mirjalili, Python Machine Learning : Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow 2, Packt Publishing, 2019.

[12] Mattias Ohlsson, Patrik Edén, Lecture Notes on Introduction to Artificial Neural Networks and Deep Learning, Lund University, 2020.

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

[14] Le Guennec A, Malinowski S, Tavenard R. Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD Workshop on AALTD. 2016.

[15] L. Sadouk, "CNN Approaches for Time Series Classification", in Time Series Analysis - Data, Methods, and Applications. London, United Kingdom: IntechOpen, 2018.