Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Multi-sensor accelerometer-based gesture recognition

Master's thesis

2020
Bc. Matej Králik

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Multi-sensor accelerometer-based gesture recognition

Master's thesis

Study program:	Computer Science
Field of study:	Computer Science
Department:	Department of Computer Science
Supervisor:	Mgr. Vladimír Boža PhD.
Consultant:	Mgr. Marek Šuppa

Bratislava, 2020
Bc. Matej Králik

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matej Králik
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Multi-sensor accelerometer-based gesture recognition
Rozpoznávanie giest pomocou viacerých akcelerometrov

Anotácia: Cieľom diplomovej práce je (nielen):

- Zostavenie vlastného prototypu rukavice vybavenej akcelerometrami.
- Zaznamenať dáta pomocou zostaveného prototypu.
- Vyhodnotiť publikované metódy na rozpoznávanie giest na zaznamenaných dátach.
- Navrhnúť a otestovať novú metódu na rozpoznávanie giest pomocou viacsenzorových dát.
- Porovnať navrhnutú metódu a analyzovať jej použiteľnosť pri viacsenzorových dátach v realite.

Vedúci: Mgr. Vladimír Boža, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 13.12.2018

Dátum schválenia: 08.01.2019 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Matej Králik
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Multi-sensor accelerometer-based gesture recognition

Annotation: The aims of this thesis include (but are not limited to) the following:

- Construction of a custom glove prototype equipped with accelerometer sensors.
- Gather gesture-based data using the prepared prototype.
- Evaluate previously published methods for gesture recognition on the gathered data.
- Propose and test a new method for gesture recognition using multi-sensor accelerometer data.
- Compare the proposed method and analyse the usefulness of multi-sensor setup in a real world scenario.

Supervisor:	Mgr. Vladimír Boža
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	13.12.2018
Approved:	08.01.2019 prof. RNDr. Rastislav Kráľovič, PhD. Guarantor of Study Programme

Student

Supervisor

Acknowledgment: I would like to thank my consultant Marek for being a never-ending stream of support and guidance. Thank you for the helpful and encouraging discussions we had. A very special thanks goes to Sára, for fully supporting me throughout this period of time. Last but not least, I would like to thank my parents for the help and support throughout my studies and life.

Abstrakt

Oblasť automatického rozpoznávania ľudskej aktivity (Human Activity Recognition – HAR), ktorá je založená na ľuďmi nositeľných senzoroch, zaznamenala v posledných rokoch zásadný rozmach. Praktické aplikácie z tejto oblasti je možné nájsť v mnohých kontextoch, najmä v zdravotníctve, či vo funkcionalite "inteligentných domácností".

V našej práci predstavujeme vlastný hardwarový prototyp WaveGlove vo forme rukavice s piatimi inerciálnymi senzormi. S použitím tohto prototypu sme zozbierali dva datasety, obsahujúce rôzne množiny giest. Tieto datasety obsahujú 1000, respektíve 10000 vzoriek.

V ďalšej časti našej práce implemetujeme viaceré klasifikačné metódy klasického strojového učenia, ako aj hlbokého učenia. Pre účely vyhodnotenia následne používame viac ako 10 verejne dostupných datasetov. Ďalej navrhujeme novú architektúru neurónovej siete, založenú na koncepte "self-attention", aplikovanú pre túto úlohu. Porovnaním dosiahnutých výsledkov ukazujeme, že táto nová metóda dosahuje lepšie priemerné výsledky, ako iné, už publikované metódy. Nakoniec tiež realizujeme ablačnú štúdiu, ktorou demonštrujeme dôležitosť nami zvolenému prístupu s použitím viacerých senzorov. Výsledkami tejto štúdie poukazujeme na fakt, že dosiahnuté výsledky sa zlepšujú s pridaním viacerých senzorov, pričom výrazné zlepšenia prestávame pozorovať pri zahrnutí viac ako troch senzorov.

Kľúčové slová: rozpoznávanie ľudskej aktivity, rozpoznávanie ľudských giest, nositeľné zariadenia, neurónová sieť

Abstract

The field Human Activity Recognition (HAR) based on wearable sensor data has grown considerably over the recent years. One can find many practical applications of HAR, especially in healthcare and smart-home control. In this work we present a custom hardware prototype called WaveGlove in the form of a glove with five inertial sensors. Using the prototype, we acquire two datasets with different gesture vocabularies consisting of 1000 and 10000 samples, respectively.

We implement several classification methods from Classical Machine Learning as well as Deep Learning. For evaluation we use more than 10 publicly available datasets. In addition, we propose a novel self-attention based non-recurrent neural network architecture, which on average outperforms the previously reported methods. Finally, we perform an ablation study on the acquired dataset, to demonstrate the importance of multiple sensors. We show an increase in performance when using up to three sensors, with no significant improvements with more sensors.

Keywords: Human Activity Recognition, Hand Gesture Recognition, wearable sensor, glove prototype, neural network, LSTM, self-attention.

Contents

List of Figures

List of Tables

Introduction

In recent years, the computing power we have access to has increased tremendously. We live in an era when computers of various sizes are integrated into things around us. Gadgets, toys, home appliances, and tools we use most of the time are a few examples of such integrations. With this rise in computing power, the ability to process greater quantities of data has a potential like never before. To gather that much data, various types of sensors are being deployed where possible. Cameras, temperature and humidity sensors, distance sensors, gyroscopes and accelerometers, radiation and smoke detectors and many others are massively fabricated and widely used.

In this work we are particularly interested in inertial measurement units (IMUs), which can be purchased for a few dollars and are present in devices used on a daily basis. In phones or tablets IMUs are most commonly used for determining the screen rotation and for gaming applications. In mechanical hard drives, they are used for protection against damage caused by a fall. In wearable devices (smart watches, etc.) they are used to determine the type and intensity of an activity the user is doing. IMUs typically consist of an accelerometer, gyroscope and a magnetometer.

An accelerometer is used for measuring acceleration. The core principle of this measurement is using *Newton's law of motion* in conjunction with *Hooke's law* for springs[1]. This allows us to calculate the applied acceleration using the distance a spring is extended. To convert this distance into an electrical signal capacitors are used. Capacitors are passive electronic components which have a property called capacitance. Capacitance depends (among other things) on the distance between the two plates of a capacitor. Therefore with the correct setup (as shown in Figure 1) we are able to measure capacitance, to measure distance and to calculate the desired acceleration.

For usage in real-world applications, the mentioned devices need to be very small. To achieve that, Microelectromechanical systems (MEMS) are used. MEMS is a technology of building devices with moving parts (like accelerometers) on a very small scale - typically tens of micrometers big. By integrating three accelerometers in directions perpendicular to each other, we can effectively measure the three dimensional acceleration.

Some accelerometer circuits are fabricated together with a gyroscope. Gyroscopes are devices used in similar cases and have alike core principles. A gyroscope is used for

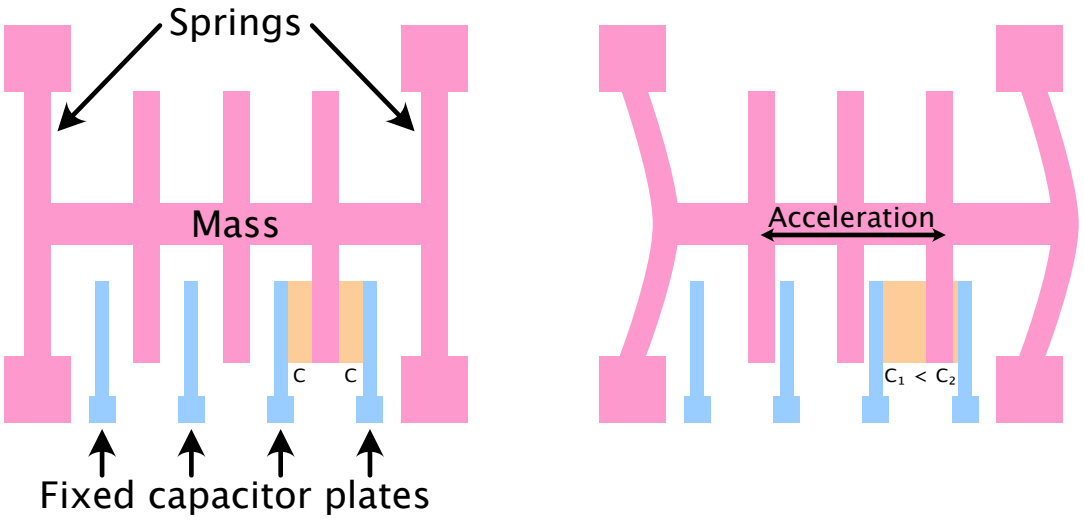


Figure 1: Structure of a MEMS Accelerometer. When an acceleration is applied to the device, the springs allow the movement of the floating plates, leading to a change in capacitance between the fixed plates.

measuring angular velocity. When an object is turned around an axis, another physical phenomenon - the *Coriolis force* occurs. The size of this force is then measured through capacitance (as with accelerometers).

In summary, IMUs are cheap and flexible sensors with a large variety of usages. One of these usages involves attaching one or more IMUs to human clothing or body and using the recorded values for classification of the activity the subject is undertaking. The area of research which focuses on this kind of classification is called Human Activity Recognition (HAR). Typically, daily activities such as walking, running, climbing stairs, sitting, or various exercises such as squats or push-ups are classified. Either a single sensor is used, which usually simulates the presence of a mobile phone in subjects hand or pocket, or multiple sensors can be used with the help of special equipment (eg. custom crafted vest). Multiple sensors are helpful in classifying activities of various body parts, but pose a rather unrealistic scenario outside of laboratory conditions. In the multi-sensor scenario, IMUs are also sometimes used in conjunction with other sensors, such as heart rate monitors or electromyograms.

A sub-area of HAR which focuses on hand gestures is called Hand Gesture Recognition (HGR). In HGR, the sensors are usually placed in or on subjects hands and are used to determine what kind of hand gesture the subject does. One of the main difference between HAR and HGR is that HAR usually works with longer time frame (up to minutes), while hand gestures tend to be short (a few seconds) in time.

In HAR and even more importantly in HGR, an important part of classification is distinguishing between states of activity/gesture and states of non-activity/non-gesture. If a sensor was attached to subjects hand for a longer period, it would be

impractical not to be able to perform anything than the predefined set of gestures. The non-gesture states are those when the subject performs no movement, or some movement which does not represent a meaningful gesture. Some of the datasets allow benchmarking this important part of classification by including non-activity data, also called as data belonging to the "null" class (each class corresponds to one of the classified activities).

There are several use cases of both HAR and HGR in ubiquitous computing - remote control, health monitoring, gaming, virtual and augmented reality, biometrics etc. More importantly, they can be useful in sign language translation[2] or helping users to quit smoking[3] by classifying hand movements.

In this work, we focus on the problem of HGR using multiple sensors, which has not been explored extensively. Most of previous HGR work focused on using mobile phones, or other commercially available products with IMU sensors, but their drawback is the lack of pervasiveness. If performing a hand gesture in order to control a home appliance involves finding a phone or other specialized device, it is hardly simpler than controlling the appliance directly.

Luckily, with the availability and price of sensors and other components, it is possible to build low-cost wearables, to assist in the multi-sensor gesture recognition. We therefore build a device prototype in the form of a glove, with an IMU attached at the middle phalanx of each finger. If such a device was built commercially, its size, weight and cost would shrink intensively, allowing for a much more pervasive experience. Using our prototype - WaveGlove, we effectively simulate this scenario and we create an opportunity to capture a multi-sensor HGR dataset to be used in evaluating various classification approaches.

To understand the core difficulty of HAR and HGR using IMUs, we need to know the limits of the sensors we use. The problem of gesture or activity recognition would be a lot easier if we could measure the acceleration or angular velocity without noise and with high precision. If the noise was low enough, or simply was not present, given the acceleration, we could compute velocity v_t and position x_t as shown in equation 1.

$$\int a_t dt = v_t, \quad \iint a_t dt = \int v_t dt = x_t \tag{1}$$

Knowing the position in time, we could apply other known methods for classification from the area of computer vision. For example, we could map the movements of a gesture onto a two dimensional plane, making handwriting, by performing gestures mid-air an easily approachable challenge.

Unfortunately, the sensor measurements are affected heavily by noise. This noise can be caused by various physical phenomena occurring during the use of our devices (eg. induction) and by our limited accuracy of measurements. Methods of denoising

the signal have been studied in the area of signal processing, but none give results plausible enough for our use case. If we were to attempt the calculation of position through double integration as shown above, we would not be successful. Given the noise levels and measurement frequency, the resulting position would diverge exponentially, or slowly drift away and no further classification would be possible.

Caused by these sensor limitations, the area of HAR still presents a huge amount of challenges to date. Various classification and segmentation techniques are researched, both from the area of classical machine learning (CML) and deep learning (DL).

In the presented work we construct the WaveGlove prototype and use it to capture a state of the art datasets which can be used for HGR. We define and adopt multiple gesture vocabularies, both to exploit the multi-sensor environment and to create comparable datasets.

We explore the various CML and DL methods used for both HAR and HGR in previous work and compare them on multiple datasets. Using our newly acquired dataset as well as 10 previously published datasets, we evaluate our classification methods. Multiple models proposed by other authors are reproduced and their performance is evaluated.

Using the acquired multi-sensor dataset, we perform an ablation study, showing both the value added by multiple sensors as well as the boundary after which adding more sensors does not further improve the accuracy. Finally, we propose a new DL classification method, which improves the state of the art on the used datasets.

In the next chapter we provide an overview of related work, both in the CML and DL areas and the importance of sensor placement and selection. Subsequently, we describe the constructed WaveGlove prototype, the used gesture vocabularies, recording framework and the overall experiment setup. In the third chapter, we outline the used datasets, their statistical properties and setup of the experiments. Finally, we compare several CML and DL classification methods using multiple evaluation metrics and schemes on the datasets.

Chapter 1

Related work

Interest in the field of Human Activity Recognition (HAR) and Hand Gesture Recognition (HGR) using inertial measurement units (IMUs) has grown noticeably during recent years as identified by [4] and [5]. Several comprehensive surveys have been previously conducted in order to summarize research and define the state of the art [4], [5], [6], [7], [8], [9] and [10]. Our work does focus on HGR, however the field of HAR is very closely related. Most notably, when recognizing an activity, the goal is to classify a long-term state that can last several seconds or minutes. Gestures operate on a much shorter timespan (a few seconds at most) and consist only of a single or a few movements. Recognizing activities is studied more intensively, as annotating long lasting states is less time consuming and usually does not require specific hardware or infrastructure setup. Longer timespans also lead to larger datasets after segmentation, which is beneficial for model training. In this work the terms gesture and activity may be used interchangeably as we refer to the performances recorded using IMUs. We will use both of these terms based on the work, dataset or method context.

We adopt the categorization of previous work from [4] into two major categories:

- Classical machine learning (CML)
- Deep learning (DL)

With the recent rapid growth in the area of Internet of Things (IoT), smart wearables and environmental sensors, the amount of HAR data available has grown dramatically. Therefore feature extraction has become a key stage in the process of the recognition.

In CML, discrimination feature vectors are designed manually or semi-manually, which requires wide-ranging domain knowledge, expert-driven decisions or guesses and other time consuming selection techniques. The major challenges of conventional handcrafted features could be summarized as in [5]:

- Manual feature extraction and design, requires expert domain knowledge.

- Selection of various features is becoming more complex and unsustainable task with the rapidly increasing amount of data and modalities.
- Feature design requires vast amounts of labelled data. Creation of such datasets is time consuming and requires a specific setup. On the contrary, IoT, wearables and other smart sensors provide a great source of unlabelled data that can be used for reinforcement learning.
- Carefully designed features from the time and frequency domains may not be able to capture the dynamic temporal dependencies in the signal.
- Ensembles and decision fusion techniques are commonly used. However, there exist uncertainties about the generalization ability of these approaches.

One of the attempts to tackle these challenges involves automatic feature extraction. This is mostly done using techniques form the are of deep learning - specifically neural networks. Deep neural networks consist of several layers of neurons interacting with each other and using activation functions to introduce non-linearity into the model. The use of DL comes as no surprise due to its undeniable success in the areas of signal processing, autonomous driving, image and object recognition, natural language processing, and various others[5][7].

As reported in [4], CML techniques still prevail over DL in the area of HAR when taking into account the amount of published papers leveraging the respective approaches. Even though deep learning methods report higher average recognition accuracy (92.8%) when compared to classical machine learning (92.3%)[4], the difference is too small to clearly determine a dominant approach. Data generation and segmentation, evaluation metrics and validation protocols have crucial impact on the resulting model performance[8]. Highly imbalanced datasets are very common in the area of HAR (most notably the ones with a null class), in which case reporting the average accuracy does not capture the actual performance of the model. A simple linear classifier which is able to distinguish between the null class and other classes may reach average accuracy comparable to that of a more complex model.

Poor quality of any of the preprocessing or evaluation stages can lead to highly skewed results as well. Other research areas, such as object detection and image classification, have already approached the problem of non-reproducibility and highly skewed results by defining proper standards[11]. In HAR however, no well-defined evaluation methods exist. More importantly, experimental evaluations[7] show that current state of the art techniques do not reach accuracy levels high enough to be used in a real world applications, even though the reported metrics may suggest otherwise.

In the following sections we present an overview of the CML and DL methods used in the field. After that, we report how sensor selection and placement affect

the recognition performance and survey previous work from this point of view. We conclude the chapter with a section focused on HAR dataset availability, usage and acquisition methods.

1.1 Classical machine learning

Previously outlined challenges of CML may present a significant drawback in comparison to DL methods. However, there are several advantages of CML as well:

- Training and testing is usually computationally less expensive. DL training in general involves complex gradient calculation and several iterations to find a global minimum. This topic is particularly interesting in embedded environment, where the speed of recognition, hardware requirements and power efficiency are very constraining.

- CML methods tend to be more interpretable to humans. Consider for instance the baseline we use in our classification: lowest distance from the mean of training data per specific class, is a very interpretable model. It comes across as intuitive, that the closest template (even when using a specific metric in a high dimensional scenario) is the most viable prediction. Methods such as support vector machines (SVM) or hidden Markov models (HMM) can be viewed as generalizations of this simple approach, using nonlinear projections or probabilistic modelling. On the other hand, neural networks – especially the deep ones – are still perceived as black-box algorithms due to their high level of abstraction.

  Explanations are especially important in situations, where making decisions was previously entrusted to humans and ought now to be automated. The area of explanatory artificial intelligence (XAI), has been studied only recently, with the increasing need to entrust important decisions to algorithms. It is also important to distinguish between the terms interpretability and completeness. In [12], the goal of interpretability is to describe the operations of a system in a way understandable to humans, while the goal of completeness is to describe a system in an accurate way. Given these definitions, deep learning may be complete but hardly interpretable. Furthermore, interpretability in DL has a slight overlap with ethical, moral and philosophical dilemmas[12].

- Hand tuning is more accessible. Especially in practical applications, where the performance in a relatively specific scenario is more important than overall generalization, hand tuning the final model is a common approach. With DL models having tons of parameters, which are hard to explain at the current state of published research, hand tuning becomes difficult to execute. On the contrary,

when deploying a solution in a realistic environment, an expert with appropriate domain knowledge is often available to guide the hand tuning of CML parameters.

- Class imbalance and annotation scarcity present a lesser problem in comparison to DL[7]. Supervised deep learning methods often require large amounts of annotated data, which is often hard to obtain in the area of HAR. As most of the datasets used in previous work are not available to the public[4], acquiring a novel suitable dataset is often required and time consuming. Unsupervised methods have been adapted to the task[5], but their performance is yet to be rigorously determined.

The process of recognising a gesture can be split into stages, creating a recognition pipeline (Figure 1.1).

The three stages which are of interest from the classical machine learning point of view are:

1. Preprocessing
2. Feature extraction
3. Classification

Methods employed at all of these three stages are discussed in the following sections. Some of the mentioned methods are still applicable and are used also in DL. Traditionally, however, they originate from the CML methods, where they play a key role in building the model, whereas in DL various hyperparameters and architectural choices have greater impact.

Stages of the recognition pipeline

After reading the raw data from the sensors, it contains noise of various types. The noise can be caused by imprecise measurement, change in environment or intra-class variability (each single human performance of a gesture will differ).

When identifying a gesture based on previous samples from the same user, we talk about **user-dependent** recognition, and we call the opposite situation when testing a gesture performed by a different person **user-independent** recognition. As shown in [13] and [14], user-independent gesture recognition poses a greater challenge, because of higher variation in performances. Variations occur not only on an interpersonal level, but performances distributed along a longer time frame (e.g. days) can differ significantly as well[15].

Some of the noise can be effectively filtered out by applying various methods in the pre-processing stage. Another challenge we may need to address in this stage is identifying the gesture itself in the stream of data. Significant body of research regarding

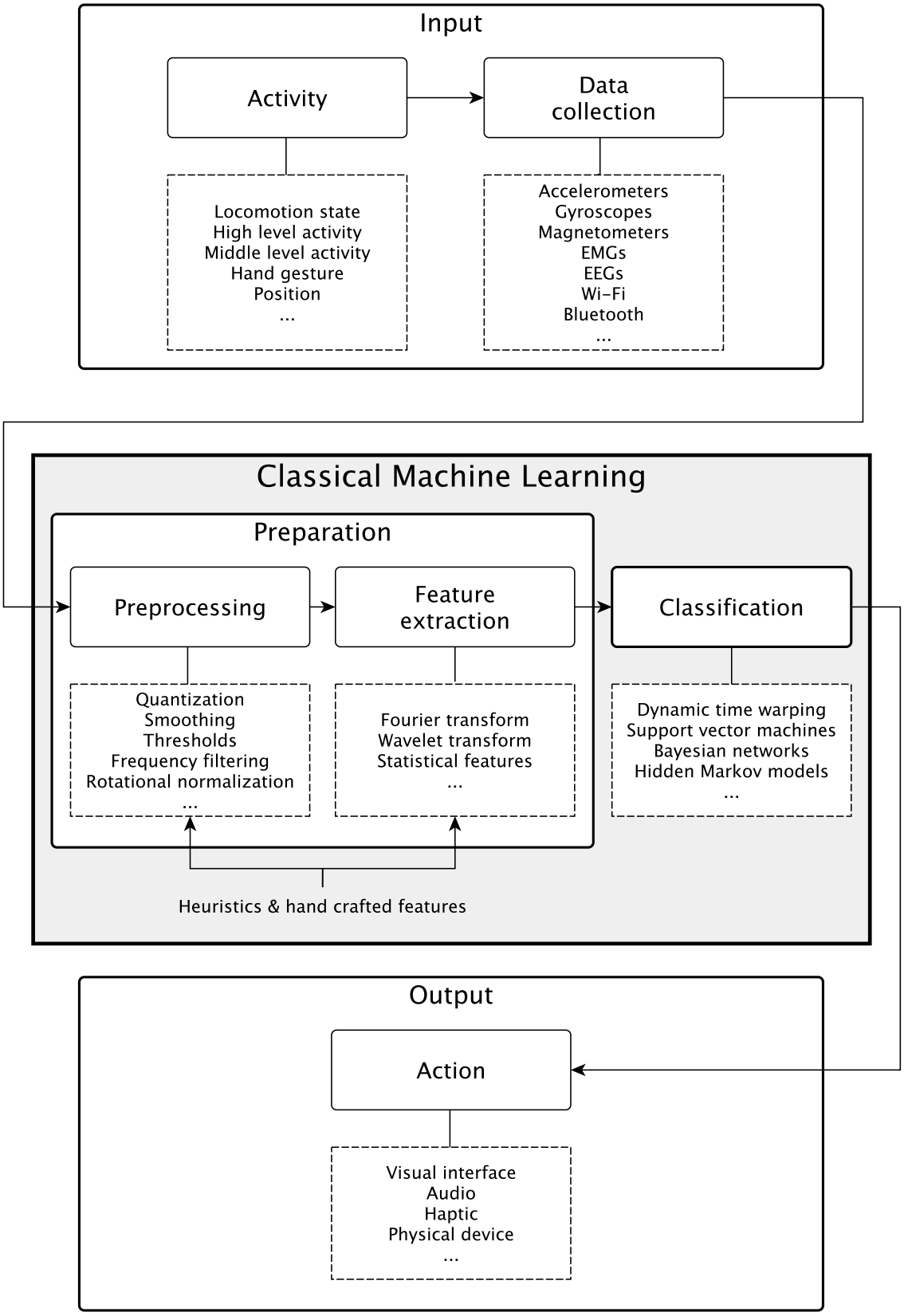


Figure 1.1: Stages of a CML recognition pipelines

gesture recognition does not focus on recognizing whether a time series presents any gesture at all, resulting in the need to mark the beginning and the end of a gesture by the user during the recording, or via other means.

As part of preprocessing, quantization can be applied, which is the process of mapping the continuous (or otherwise large) set of values to a smaller, discrete set.

After pre-processing and quantization is applied, features may be extracted from the data. The corresponding model then works on these features instead of the original raw data.

The next stage is finally responsible for the classification itself. A trained model receives the data or some features as an input, and associates it with one of the predefined classes.

The recognition pipeline stages serve only as an approximation. There are several approaches, which leave out some of the stages. These include: working with the raw data directly, without feature extraction[16], merging multiple stages into a single process, or adding further stages, such as custom model adjustment[15].

1.1.1 Preprocessing and quantization

The goal of pre-processing is to transform the data into a simplified or compressed form, to allow better recognition. This transformation typically tries to remove some of the noise, or other redundant parts of the data. Each of the following pre-processing techniques can be considered as a heuristic trade-off, based on a belief that it removes the part of the data which was likely unnecessary for the classification. The appropriate constants and thresholds for such heuristics are often determined by guesses and experiments. In the case of simple hand gestures, they might not contain a lot of information, but when classifying a longer sequence of a more complex activity in a multi-modal environment (e.g. cycling), the amount of information can be significantly higher. Due to the physical limitations and realistic influences however, all sensor recordings contain inherent noise and other forms of redundant data. This leads us to the effort of removing this noise and redundancy, to further reduce the amount of data pushed into our models, leading to simpler, faster and better training and inference.

Fixed set quantization

Instead of directly using the measured values from a sensor, which are usually 10 to 20 bits in precision, we can exploit the fact that a human can consciously perform movements only up to a certain precision.

The most straightforward approach would be to map the range of values from the input values onto a smaller scale (e.g. 5 to 8 bits). During quick acceleration a human has less control than during slower movements, which intuitively leads to a non-linear

scale. This non-linear quantization is tested by [15], where 33 levels of acceleration are used. Only two of the levels are used for values above $2g$[1] (and below $-2g$) while 10 levels are used for values from $-2g$ to $-1g$ and $1g$ to $2g$ and the remaining 20 levels are used for values between $-1g$ and $+1g$. The remaining level is used for $0g$ values (seen mostly in sample padding).

The utilized quantization approach does not have to be axis-independent like the previous example. If we suppose, that the user consciously performs movements only in a few standard directions (e.g. the three major axes and their 45° rotations), we could map each triple of consecutive acceleration (or angular velocity, or orientation) values to the closest point on a sphere, ellipse or similar three-dimensional object. This quantization is used in [16] and [17] with mappings to various sets of points.

Sliding window smoothing

Hand gestures tend to be rather smooth by definition – a circle, or even a set of movements in different directions are smooth most of the time. A common way to reduce spikes and smooth out a signal is using a moving average filter [18]. The average filter replaces individual values with the mean of these values from an interval centered at given value. This creates a sliding window, which is moved forward by a constant after each calculation (typically smaller than its size).

The resulting signal tends to be smoother, due to the sudden changes in values representing only a small weight in the mean. Moving filters can calculate arbitrary functions on the windows, like weighted mean (where higher weight is closer to the center), and they are widely used in digital signal processing as well as in other areas, such as finance. This sliding window technique is used in gesture recognition as well[15][19][20][21]. In Figure 1.2, we show the raw x-axis acceleration from a circle gesture performance and the signal after applying moving average filter.

Various thresholds

Thresholding can be used either during the task of identifying a gesture or significant activity from a continuous stream, or when trying to reduce the length of model input. The rationale is that hand gestures could be defined by those moments, when the user makes a significant change in acceleration or direction of the movement. By filtering out the data which does not contribute to these moments, we can remove unnecessary parts of the data.

To remove passive parts of the data, it is common to use a simple threshold for the amplitude of acceleration ($a_t < \epsilon$) [16][22][23]. The constant used for this truncation can also depend on properties of the data, like standard deviation [21]. To filter out

[1] g refers to the gravitational constant - approximate gravitational acceleration on Earth.

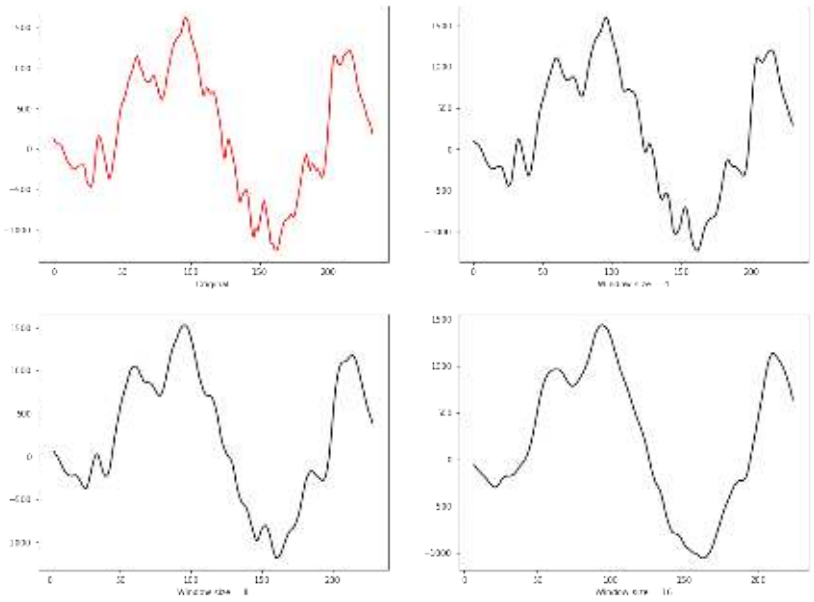


Figure 1.2: Moving average filter - the original signal (top-left) and sliding window smoothing applied with window sizes of $4, 8$ and 16.

frames which do not change the direction significantly, we can compare the components of each acceleration vector with its predecessor ($a_{t-1} - a_t < \epsilon'$) as in [16].

When we truncate/filter our input data as we just described, the sequences will vary in length (as they likely did before the truncation). If the considered classification model requires data of constant length, some kind of interpolation or padding has to be applied. One example would be cubic spline interpolation[20], in general however, zero padding the data is a common approach. Examples of using the thresholds and interpolation are in Figure 1.3.

Frequency filtering

A different approach, common in signal processing[18], is to use features from the frequency domain instead of the time domain. The reasoning behind this transformation is that the human movements characterizing the gesture have relatively small frequencies compared to those introduced by noise. The solution is to apply a low-pass filter to remove the noise[22]. An example usage of this filter in Figure 1.4 and Figure 1.5 shows the spectrum of the acceleration data from a recorded hand gesture instance.

Alternatively, the frequency domain coefficients can be used directly as an input to the classification model[24]. This simulates low-pass filtering, because only a portion of the lowest frequencies with high amplitudes is considered.

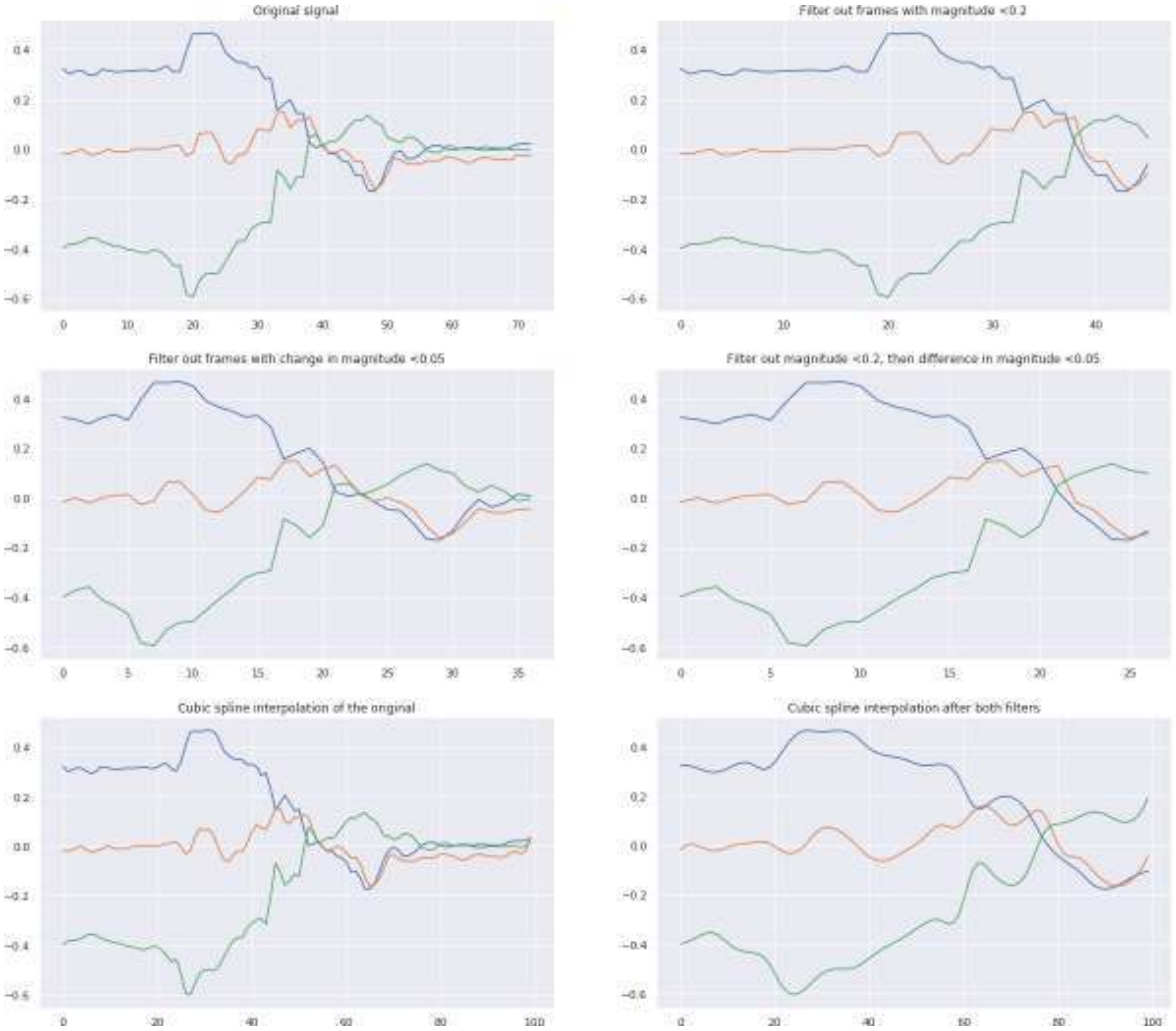


Figure 1.3: Signal of a *Page flip* gesture from an accelerometer on the middle finger with various thresholds applied (note the different lengths of the presented sequences).

Rotational normalization

When designing a gesture recognition system, one must decide whether the same gesture rotated into a different plane should be considered as an identical or different gesture. In the case of ubiquitous and immersive sensor presence, the user might want to perform low-level gestures, such as finger movements in different high-level postures – standing, lying, crouching. Employing rotational normalization techniques may therefore improve the recognition accuracy at the cost of reduced gesture space.

One of the approaches is to plot the values of acceleration, angular velocity, rotation, etc. as points in three-dimensional space and normalize their rotation with regard to a best fit plane[22]. Furthermore, after the best fit plane is found, all the points can be projected onto it, and a best fit line on this plane can be used for rotational normalization again[23].

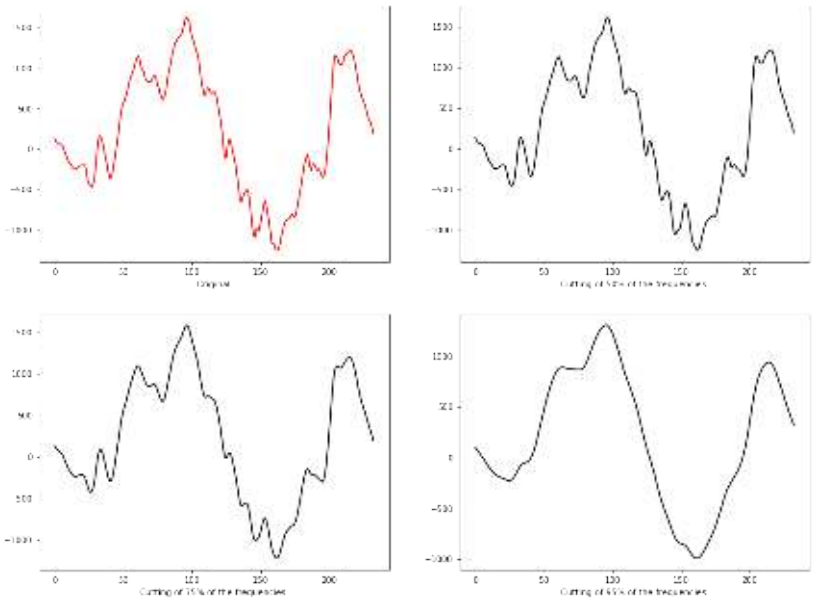


Figure 1.4: The original signal and the low-pass filter applied on relevant portions of the frequency spectrum.

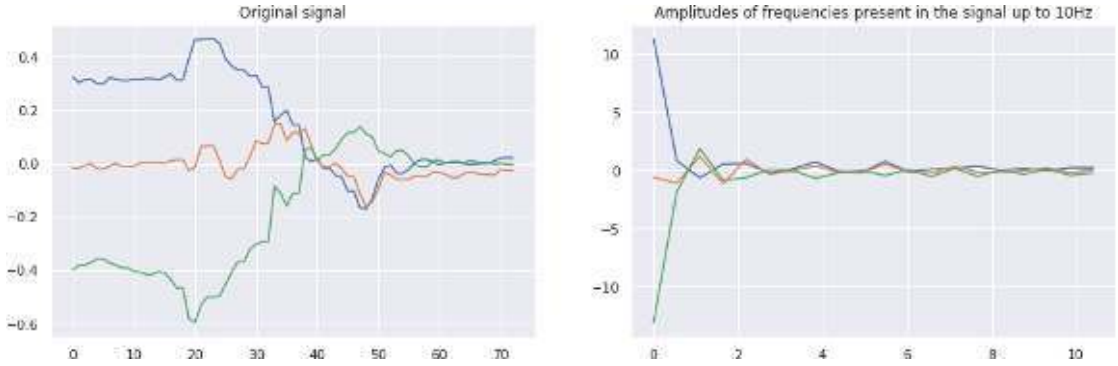


Figure 1.5: Signal of a *Page flip* gesture from an accelerometer on the middle finger and the corresponding amplitudes of frequencies up to 10Hz. The relation demonstrates that high frequency movements are unlikely to appear in a hand gesture.

1.1.2 Feature extraction

Based on the method we use in the gesture classification stage, working directly with the time series data may not be acceptable. In such cases the input data is transformed into a different domain, or specific statistical features of the data are extracted. Mean, correlation or standard deviation of the channels in the input data are examples of statistical features that can be used [14].

In general digital signal processing, data is transformed for many purposes (not only) into the frequency domain[18]. In the subsections below, we describe transformations used in previous work. A great deal of other transformations is already being utilized in different fields, which might be successfully applied to human activity

recognition or human gesture recognition as well.

Fourier transform

Being one of the most used time-to-frequency domain transformations, it is no surprise that Fourier transform (in its discrete form) finds an application in gesture recognition as well [24][21]. The output of this transformation gives us information on the amplitudes, frequencies and phase offsets the input is composed of.

Most library implementations allow us to specify a precision up to which we wish to calculate. The higher precision we choose, the more frequencies we decompose our signal into. As noted in the subsection about frequency filtering, the lowest frequencies tend to characterize most of the gesture, and are of interest for feature extraction.

Wavelet transform

Decomposition into sines and cosines (e.g. Fourier transform), has several disadvantages when processing data containing sharp spikes. The problem originates in the fact that sines and cosines are non-local functions spreading out to infinity, leading to a poor performance on data containing sudden changes [25].

This problem can be solved by using a slightly different class of functions for decomposition. Applying wavelet functions, scaled both in amplitude and frequency as described in [25], can help us represent sharp spikes in a very sparse way. One of dissimilarities between the Fourier and Wavelet transform is the fact that wavelet transform is specified for a whole class of so-called wavelet functions.

For various applications, the choice of the wavelet function may have a great impact on the resulting performance. In [13] and [26], the Haar wavelet and coefficients of its transformation are used.

Other transforms

The decomposition methods used in other fields provide a vast, yet unexplored possibilities. Discrete cosine transform was explored in [24], however no extensive study has been made on different decomposition methods and their usage in gesture recognition.

1.1.3 Windowing

Especially in the case of HAR, where the recorded activities can be several minutes long, it is not feasible to classify such long continuous streams. From the practical point of view it would be infeasible for a system to recognize a long lasting high-level activity (e.g. running) only after it finishes. To resolve this issue, the signal stream is segmented into windows, typically of a fixed length. These windows can

also overlap, which further increases the dataset size - a desirable side effect in many instances. The specific method used to create windows that are used for classification or their overlap can have tremendous impact on final model performance and can lead to skewed experiment results if not used properly[8].

Segmented windows can then be used for further transformations. For example in [14], the mean, energy, entropy, standard deviation and correlation between consecutive windows are calculated and used as features. The rationale behind this approach is that these features could reasonably characterize the motions in the gesture or activity.

1.1.4 Classification

Efficient preprocessing and robust feature extraction are the necessary prerequisites of a successful recognition, but the core complexity and the key to success still lie inside a proper classification model.

One of the important properties of each model is the number of training samples does it require to operate efficiently. When attempting a user-independent gesture recognition, the model can require a large set of training data and computationally intensive training. In such case, both the training and data acquisition can be executed up-front in a controlled environment. Running the model in final production should not require immense computational power, so that it can be carried out on an average mobile device or deployed to an embedded platform. Another desirable property of a model is when it is able to adjust itself over time, based on processed data. These adjustments should also require less computational power than retraining the whole model.

User-dependent gesture recognition, seems to achieve greater results[14][27] because of lesser variations between performances of the same gesture or activity (intra-class variations). To increase practicality of a user-dependent model, it should require a far smaller set of training data. If a user is required to perform a gesture only a few times (or zero times) for the model to start successfully recognizing it, it is sufficient for pervasive usage.

Baseline

The most basic model we could use, would be to naively use a metric (euclidean distance, cosine distance, etc.) to compute the distance between the input data and the saved templates labelled for the considered gesture classes. The class with the smallest average distance of samples in the training set from the classified samples, would be then chosen as the result of classification. A simple model like this would have plenty of pitfalls, but it serves as a good example. In [28] a very similar simplistic technique was shown to have better than random, but still not feasible results.

When coming up with a more appropriate model, one way to think about it is to ask how do the performances of a single gesture differ. If we tried to perform the same gesture twice, we would inevitably fail at executing each of its parts at the same speed again. Below, we present dynamic time warping (DTW), a common algorithm used in signal processing, to address timing inconsistencies in time series data. After that we discuss various CML methods used in prior work.

Dynamic time warping

To account for possible time inconsistencies in individual performances of a gesture, we will simulate the ability to arbitrarily extend and contract parts of the performed gesture in time. The final distance between two gestures will be represented by the minimal sum of differences after extension/contraction. Dynamic time warping (DTW) is a widely used algorithm for signal processing, with plenty of heuristic improvements [29]. Dynamic time warping can be fully implemented through the usage of dynamic programming and is widely used in accelerometer gesture recognition [15, 19, 22, 30, 31].

To find the DTW distance, let $X = \{x_1, \dots x_n\}$ and $Y = \{y_1, \dots y_m\}$ represent the signals and let

$$D[i][j] = |x_i - y_j|, \forall i \in \{1, \dots n\}, j \in \{1, \dots m\}.$$

To calculate the minimal matching $A[N][M]$ we can use the program shown in 1.1.

Listing 1.1: Implementation of dynamic time warping

```
A = Matrix(0...N × 0...M)
A[0, *] = A[*, 0] = ∞
for row in 1...N do
  for column in 1...M do
    A[row][column] = D[row][column] + min(
                     A[row][column - 1],
                     A[row - 1][column],
                     A[row - 1][column - 1])
```

The time complexity of this algorithm is $O(n^2)$ and the space complexity can be improved to $O(n)$ by remembering only the current and previous row.

Under certain circumstances the time complexity of $O(n^2)$ may not be acceptable due to constrained environments. A simple way to narrow the search for the best matching is to present a constraint on the amount of extensions or contractions of the signals. These constraints will prevent us from drifting away to an excessively deformed matching. Two commonly used constraints are Sakoe Chiba band[32] and Itakura parallelogram[33][29] (Figure 1.6 shows implemenations from the pyts library[34]).

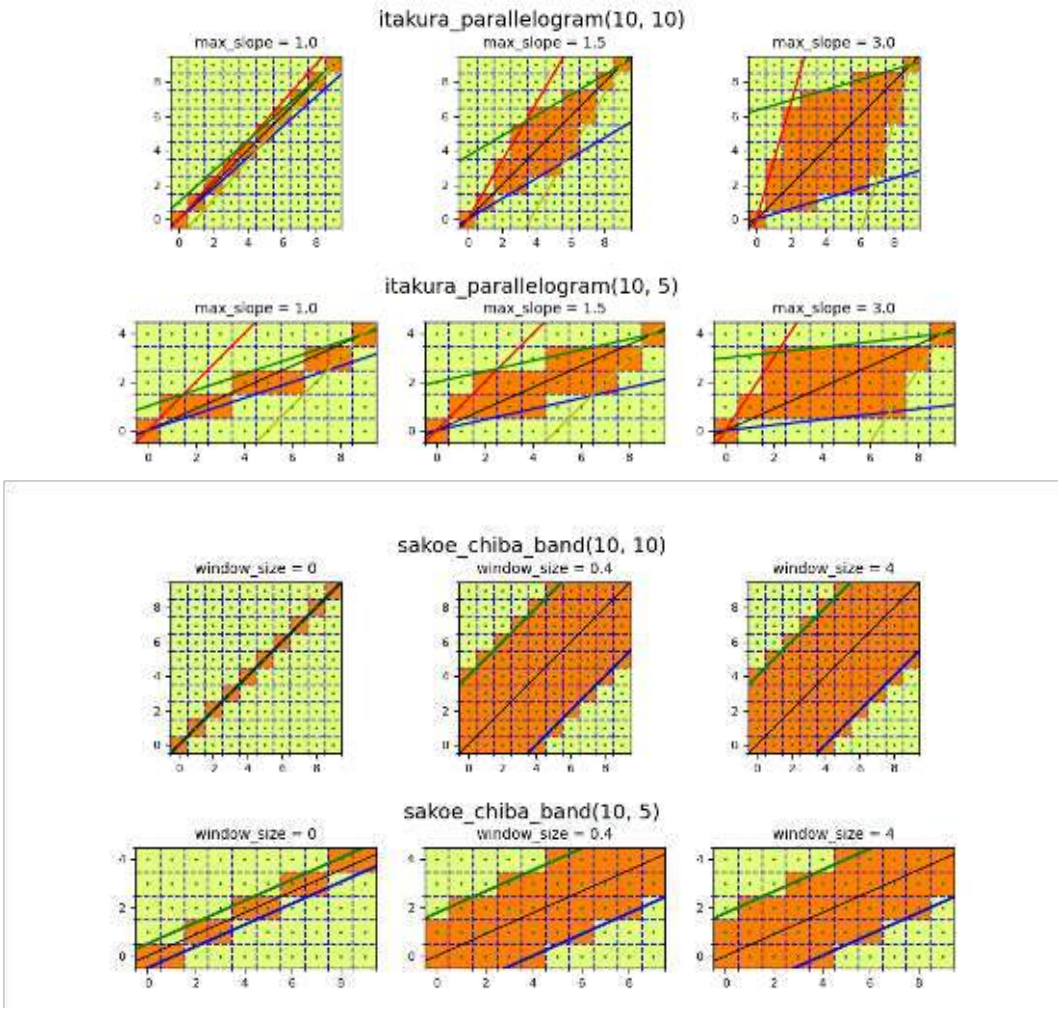


Figure 1.6: Itakura parallelogram and Sakoe Chiba band as implemented in the pyts library[34]. Both of these techniques constrain the matching of the two signals, that can be found by the algorithm. The orange highlighted fields mark the positions which can be matched.

A more flexible alternative presented in [29] is called FastDTW. This modification of the algorithm brings time complexity down to linear, with a configurable parameter that determines the possible error margin. By recursively projecting a coarse DTW solution into multiple levels of refinement, the algorithm is often able to obtain competitive results.

Hidden Markov models

Hidden Markov models (HMM) are a widely used probabilistic method for time series modeling[35]. Usually HMMs in gesture recognition try to model the most likely performed gesture using pre-calculated statistical properties of the training data.

This recognition method assumes that the examined time series conform to the

Markov property (of first-order Markov processes), which states that every state of the system depends only on its direct predecessor state. This may seem rather unintuitive, but HMMs were successfully used in [17], [16] and [36] with relatively high accuracy when compared to other classification models. Similar to HMMs, Naive Bayesian modelling has been used in [22].

Support vector machines

Finding a separating hyperplane in a high dimensional projection with the greatest minimal distance to any of the training data is the key idea of support vector machines (SVM). Using non-linear kernels functions, SVMs can map input data to higher dimensions in a way that allows for non-linear separation in the input dimensions [37].

Using SVMs requires meaningful feature extraction from the input data. Unsurprisingly, SVMs present the most competitive CML classification model used in gesture and activity recognition [13][14][24][21][38].

1.1.5 Classifier ensembles

Reducing classifier variance and providing better generalization and robustness can be achieved using various classifier ensembles. Typical approaches include bagging[39], decision forests[40], random forests[41], boosting and various voting approaches. In an ensemble, results several classifier of the same or different types are combined to form a final prediction.

In HAR, ensembles were explored in [42], [43] and [44]. Most notably [45] used a majority voting ensemble composed of logistic regression, multi-layer perceptron, SVM, K-nearest neighbour, Gaussian naive bayes and random forest classifiers. Use of the ensemble has lead to improving evaluation metrics (and their confidence intervals) over previous work using similar techniques[46].

1.2 Deep learning

In recent years, the area of deep learning turned out to be widely embraced in multiple research fields. Surveys [7] and [5] report that the fusion of human activity recognition and deep learning has began to rise only recently. Few of the reasons, why DL not replace CML in the area of HAR and still faces the reluctance of research, are lack of theoretical support, need for large dataset, long and computationally intensive training and hard optimization.

The most attractive property of deep learning is its multi-layer structure which allows learning of high-level, abstract features (effectively replacing the manual feature engineering). With the help of the latest computing resources such as GPUs and

TPUs, the high parallelizability of deep learning models leads to outstanding learning ability. Various neural network architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and other mechanisms such as attention create a multitude of possibilities on how to approach the task at hand. Several generative models are also used in the field such as stacked autoencoders, Deep Boltzmann Machines (DBM) or Deep Belief Networks (DBN). In our work, however, we focus on the "traditional" deep neural networks as they have been reported as having promising performance in recent work.

Neural networks

Deep learning refers to a broader set of techniques, from which the indisputably most popular are Deep Neural Networks (DNNs). The book [47] provides an introduction to the topic of neural networks on top of which we build the following paragraphs.

The basic building block of a neural network is a neuron. Neurons can be of multiple types, but the simplest is a perceptron (Figure 1.7).

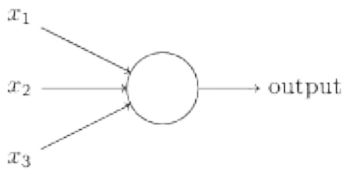


Figure 1.7: A single neuron with three inputs and one output.

Each neuron receives a set of inputs $x = (x_1, \dots x_n)$ and has a weight vector w. In the case of a binary perceptron, it emits an output $y = max(0, sgn(w \cdot x))$. This effectively means, that a single perceptron uses its weight vector to create a linear combination of the input x and returns 1 if the result is positive and 0 otherwise. Neurons usually also have another property - bias b, in which case the output is $y = max(0, sgn(w \cdot x + b))$. For the task of binary classification, a neuron like this could be used, and if there exists a separating hyperplane of the input space, then there exist w and b such that the neuron achieves absolute accuracy ($w \cdot x = b$ would be the equation of the respective hyperplane).

A group of these neurons can form a layer of neurons and multiple layers form a network. Output of each layer serves as an input to each of the neurons in the next layer, as shown in Figure 1.8. Using these connections, more complex models can be created.

In practice, the operation of calculating the output vector y of a single layer of neurons, is implemented as a matrix multiplication - $y = f(Wx + b)$, where W is the matrix consisting of the w vectors of the neurons in the layer and f is called an activation function. In the case of the perceptron the activation function $f(x) =$

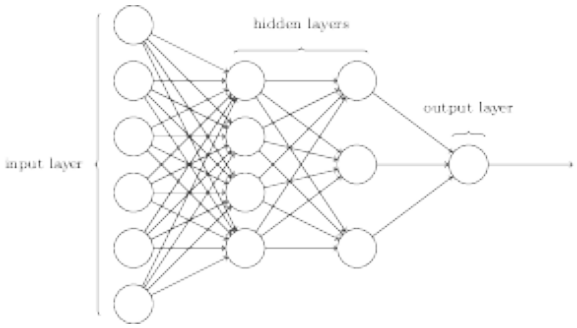


Figure 1.8: A simple neural network with 6 inputs (the input layer), two layers with 3 neurons and a final layer with one neuron.

$max(0, sgn(W \cdot x + b))$ and is called the step function, but in practice different activation functions are used. An important property of this activation function is that it is non-linear, otherwise, the calculation of y would be a linear projection. Due to the transitivity of linear projections, the multiple layers of network would also form a linear projection and hence lose discriminative power. The three most widely used activation functions are Rectified Linear Unit (ReLU), sigmoid and hyperbolic tangent (Figure 1.9).

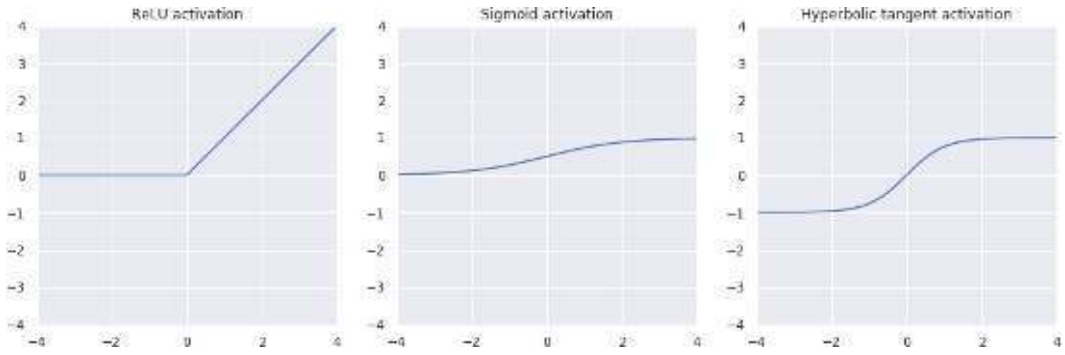


Figure 1.9: Rectified Linear Unit (ReLU), sigmoid and hyperbolic tangent activation functions.

In order to make the defined neural network model usable, we need to perform training. Training refers to the process of using a labelled set of data (inputs and expected outputs) to determine the weights W and biases b (parameters) for the neurons in the network. The training happens by iterating over epochs. At the beginning the weights and biases are usually initialized randomly. After that, during each epoch, every input x from the training set is fed to the network, yielding an output $\hat{y}$. For the network, we then define a cost function $C(W, b)$ which will denote "how good" were the predicted outputs from the training set. Various cost functions can be used, but their main property is differentiability (a sum of mean squared errors between y and $\hat{y}$ could be used). The cost function is usually a sum of the so-called loss function values over each pair of $(y, \hat{y}$. We can then define the goal of the training to minimize the loss

function over the whole training set.

Using the gradient ∇C, we can calculate the partial derivatives $\frac{\partial C}{\partial w_i}$ for every weight in the network and $\frac{\partial C}{\partial b_i}$ for every bias in the network. In layman's terms, these partial derivatives correspond to the slopes in the high-dimensional space at the point corresponding to current parameters. Therefore at the end of each epoch, these slopes can be used to adjust the weights and biases to "get closer" to a local minimum of C:

$$\forall i : w_i' = w_i - \alpha \frac{\partial C}{\partial w_i}$$

$$\forall i : b_i' = b_i - \alpha \frac{\partial C}{\partial b_i}$$

This method is called gradient descent and the parameter α determines "how fast we approach the local minimum". Of course, the local minimum may not correspond to a global minimum, which would be more plausible to find, therefore various approaches can be taken to update weights. If the weights are updated after processing each sample, we call the process stochastic gradient descent. Various approaches to descending the gradient and updating the weights can be taken as shown in [48]. The different methods used for updating the weights are called optimization algorithms and arguably the most popular one and the one we use is Adam[49]. The process of adjusting weights in two dimensions is visualized in Figure 1.10.

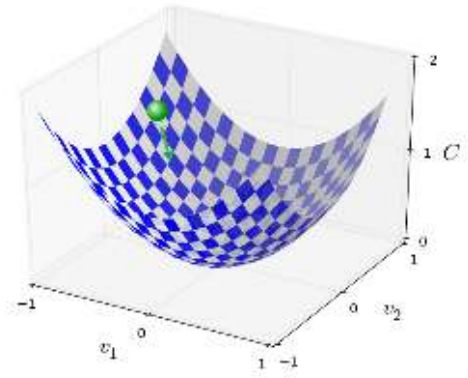


Figure 1.10: The squared surface shows the values of the cost function C and the green ball shows the current values of weights. The green arrow corresponds to the direction in which the weights will be changed after applying the gradient based adjustments.

To summarize, a neural network consists of layers, which consist of neurons. Each neuron has a set of weights (and a bias). To calculate the output of the neural network a set of matrix multiplications and non-linear functions is applied onto the input. The weights are initialized randomly, and during the training of the network a loss function is used to iteratively adjust the weights "to produce better results next time".

Neural networks are typically used to predict either a single continuous value (regression) or to determine a target class (also called label) for the input (classification). The output of a network performing classification one-hot encoded, meaning that a single neuron on the output layer represents each of the classes. The neuron with the highest value determines the result of classification. During classification the softmax function can be used, which normalizes a vector of real valued numbers into a probability distribution. That is, after applying the softmax function the values of the vector add up to 1 and are from the interval $(0, 1)$.

In practice, the neural network layers we defined above are called linear, fully-connected or dense, because all the neurons in the layer receive all the inputs and pass their outputs to all the neurons in the next layer. Over the course of years, a multitude of various neural network architectures was proposed, comprising of several different types of layers, activation functions, or other architectural features. The most common and popular ones are described in the following subsections.

Building a network from layers, defining the layer interactions, activation functions and optimization algorithms can be done with the aid of modern open-source deep learning libraries. The training process, gradient calculation and updating the weights can be computationally intensive with large datasets. To improve the speed of training, the data may be split into batches (typically sized from 32 to 512) and calculations for the samples in the batch are carried out in parallel on a GPU or TPU. Updating the weights in the network is then done for the whole batch "at once". In our work, we use the PyTorch[50] library to build the networks.

1.2.1 Convolutional networks

One of the key benefits of neural networks over CML methods is the ability to automatically extract features from the input. To further aid this process Convolutional Neural Networks (CNNs)[51] use a (typically rectangular) receptive field of a convolutional unit, with a weight matrix (Figure 1.11).

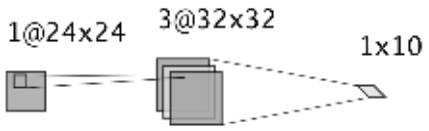


Figure 1.11: Convolutional neural networks are usually used with images. A grayscale input image of size 24×24 is fed through a convolution layer. The receptive field (kernel) is depicted as a square over the input layer. A single weight matrix of the size of the kernel is trained for each of the three channels in the next of the layer.

The rectangular receptive field is gradually attached to the input values of the layer,

and for every attachment a new value is calculated by multiplying the input value by a weight matrix and usually also applying a non-linear function. Conceptually, the receptive field (also called kernel or filter) can have an arbitrary number of dimensions. One dimensional kernels tend to be used on time series data and two dimensional kernels are typical in the area of image processing (in this terminology a two dimensional kernel actually has three input dimensions - width, height and depth typically refers to color channels in images). Multiple kernels with different, separately trained weight matrices (gradient is calculated for each of them) can be used to create more than one output channel. Each of the kernels "convolves" values from rectangular areas of the input (from all input channels) into a single output channel.

An example convolutional network used for HAR is illustrated in Figure 1.12. The input from five sensors with six axes of temporal length 100 is represented as a single channel image. In the first convolution layer, 18 filters of size 6×11 are applied. Following the convolution layers, a max pooling operation is applied. Pooling operations are a way to downscale the layers while pertaining meaningful information. The pooling functions work on a set of inputs (typically from a rectangular area of the input, similar to convolutional layers) and produce only a single output value. The most used pooling functions include computing the maximum or average of the rectangular area of inputs. For example, if max pooling with kernel of size $(3, 4)$ was used on a layer of size $(24, 24)$, the size of the result would be $(8, 6)$. A set of convolution layers is typically followed by an intermediate fully connected (dense) layer and a final dense output layer completes the network. The final dense output is most desirable during the task of classification, as it allows one-hot encoding of the output.

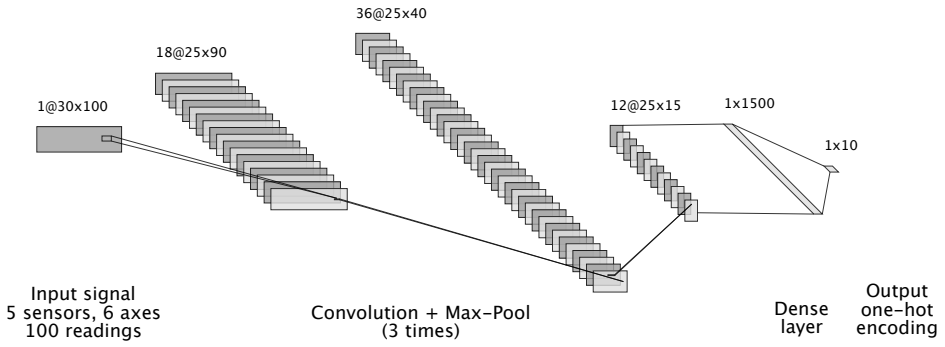


Figure 1.12: Convolutional neural network architecture.

CNNs were successfully applied to the task of HAR in previous work. In [52] an architecture of three convolutional and max pooling layers followed by a fully connected network was used, with results suggesting that their use has great potential in the area. Convolutional layers were also used as part of the network architecture in [53], [54] and

[8] reported their use in other work as well. When using a two dimensional convolution with inputs from heterogenous sensors on one of the convolved axes, data from different sensors will be fused together, which may not be always appropriate. Additional rows of zeros used as padding can be inserted, effectively preventing a kernel overlap between sensors (modalities)[8].

1.2.2 Recurrent networks

Neural network architectures which were discussed up until now (also called feedforward networks) are limited by interpreting the whole input "as one", possibly not being able to learn temporal dependencies. To break this assumption and allow more advanced time series modelling, the recurrent neural networks (RNNs) can be used. RNNs add additional recurrent connections to the neurons - the activation value of the neuron from a previous time point is fed back to itself, along with its standard input (Figure 1.13).

In the domain of HAR the dimension along which the recurrence happens is time. RNNs do not require a constant length input (when using other architectures, inputs are typically zero-padded to conform to a constant length). Self-fed activation can be viewed as a memory cell, which can hold data about previous observations, possibly allowing the framework to learn temporal dynamics of the data.

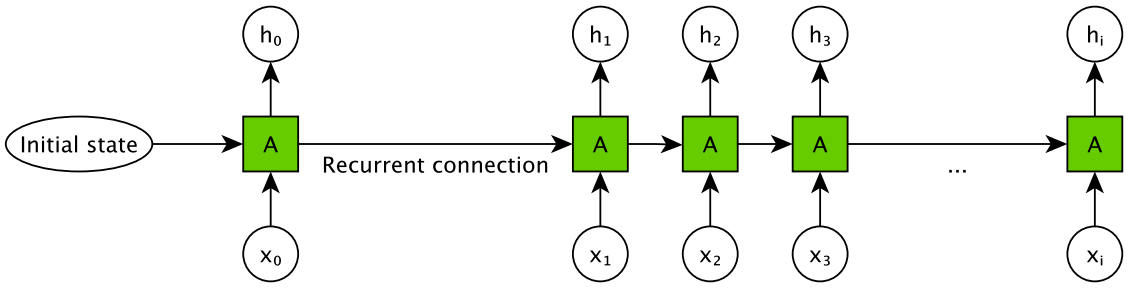


Figure 1.13: A recurrent layer receives both a standard input and an output from itself in the previous time point.

The above-mentioned simple recurrent memory cells successfully demonstrate the concept of recurrence, but are rarely used in practice. Their weak point lies in modelling long term dependencies. In theory, the memory cell could be used to pass information through a longer temporal window and we could even manually construct such weights that would achieve this effect. Unfortunately, using gradient descent in practice leads to fundamental difficulties in learning these long-term dependencies[55].

The size of the calculated gradient during back-propagation of the loss is the reason of these problems. For certain activation functions some values have very small

gradients (values away from 0 in the case of tanh and sigmoid) or the gradient is equal to 0 (negative values in the case of ReLU). This phenomenon leads to very slow or no training and is called vanishing gradient. If an activation function with areas of high gradient values is used, one risks encountering the opposite problem - exploding gradient.

The most commonly used approach to prevent the vanishing gradient are Long Short Term Memory networks (LSTMs). LSTMs are a specially designed kind of RNN, which is more capable of learning long-term dependencies. The key to their success is that the cell state (upper of the two inputs and outputs in Figure 1.14) is mutated only using a linear chain of operations, which allows better gradient propagation during learning. LSTMs have shown to be more effective in various tasks, such as speech recognition[56] and machine translation[57] than plain recurrent networks.

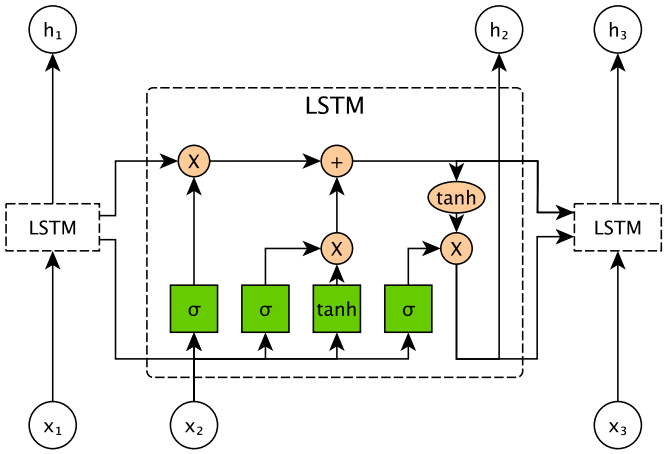


Figure 1.14: Long short term memory cell (LSTM) inner structure. Green rectangles represent layers, which have their trained weights and are followed by the corresponding non-linear function. Circular shapes represent point wise operations. Each cell has two inputs and two outputs which are initialized to zeros.

Training an LSTM requires arguably higher amount of parameters, leading to higher computational costs. Each of the four rectangular layers in Figure 1.14 requires its own set of weights, which is separately trained. The leftmost of the four trained layers is also sometimes called the forget gate, as it simulates calculating information that we might want to "forget" from the cell state in the following multiplication. The second of the layers, which also applies a sigmoid activation function, is called the input gate. Its function can be viewed as to decide which values are going to be updated in the cell state. When multiplied with the output of the third cell, the resulting state update is added to the cell state. Lastly, the output gate is a final sigmoid layer, which is multiplied with the cell state put through hyperbolic tangent. The final output serves as an output of the cell itself as well as an input for the next operation in the LSTM

chain.

RNNs and LSTMs in particular were widely used, often in combination with other techniques, such as convolutional layers in previous work. A single LSTM layer followed by a softmax layer was used in [58] and has shown very promising results in online gesture recognition.

Ensembles of LSTM networks were introduced in [59]. Epoch-wise bagging of the outputs of LSTMs was used during training and the validation set was used to select a subset of the trained LSTMs which performed the best. This subset formed an ensemble evaluated on test data to form final results by fusing the prediction of the ensemble members.

Bidirectional LSTM (BiLSTM) layers consist of two LSTM layers, with the first working as a standard LSTM layer and the second being reversed in the time domain bringing the possiblity to propagate information both in left-to-right and right-to-left contexts. In [20] BiLSTM layer followed by average pooling is used with a specialized loss function.

LSTMs and CNNs can be combined into Convolutional LSTM networks (ConvLSTMs) as shown in [60]. ConvLSTMs seem to capture spatiotemporal correlations better and were shown to consistently outperform pure LSTMs on the task of precipitation nowcasting (local, short period rainfall intensity prediction). ConvLSTMs also use additional "peephole" connections[61] which add cell state as an input to the three sigmoid (forget, input, output) gates. Most importantly, as shown in Figure 1.15, convolutions are used in the place of basic learned weight matrix multiplication on the input of the gates. ConvLSTMs were successfully used for HAR in [62], using an architecture consisting of five layers - ConvLSTM, MaxPool, ConvLSTM, LSTM and a final fully connected layer operating on the output of the last cell of the previous layer.

DeepConvLSTM[53] consists of four convolutional layers, followed by two LSTM layers and a final softmax normalization. It was shown that the two LSTM layers lead to better performing network than two fully connected layers in their place. Both of these networks also outperformed other CML methods evaluated on multiple datasets. Similarly, [54] used a convolutional network, followed by a fully connected layer, whose outputs were then fed to an LSTM layer and a final fully connected layer and softmax normalization.

1.2.3 Attention mechanism and transformers

Even with the use of LSTM, with sequences that are long enough, effective training may be problematic. Traditionally, the output of the last LSTM cell is only passed on to the next layer in the network. Therefore information from the beginning of a

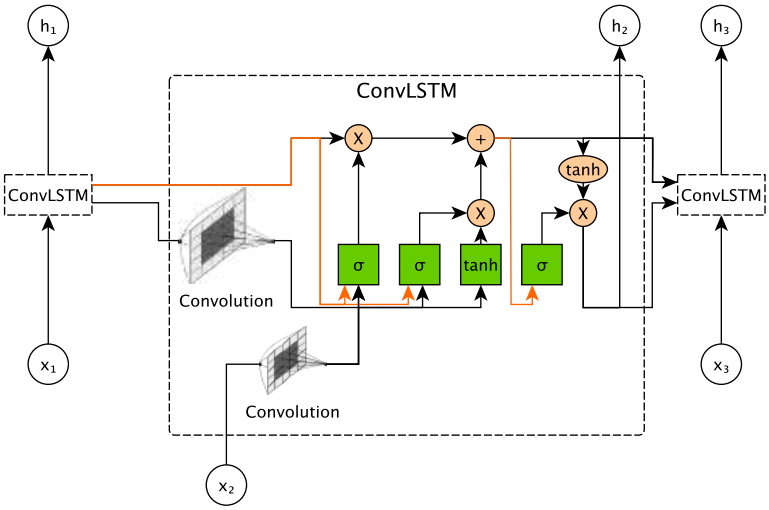


Figure 1.15: ConvLSTM cell inner structure. Orange arrows represent the newly introduced "peephole" connections, which add the cell state as an additional input to the gates. The two convolution operations allow further information propagation.

sequence needs to pass through a lot of connections to have an impact.

In [63], the authors proposed using a mechanism called attention, successfully used in the field of neural machine translation. In the context of LSTMs, attention may be used to calculate "attention scores" for the outputs of the LSTM cells. These attention scores (after softmax normalization) may serve as weights for the outputs of the cells.

There are several methods on how to calculate the scores and how to use them to combine the cell outputs. In the simplest case, the scores can be learned using a learned weight matrix and combination is done using weighted mean.

More recently, [64] introduced a novel architecture - the Transformer. Transformers are based solely on the attention mechanism, possibly superseding RNNs, by getting rid of the need for sequential calculations. Setting the new state of the art standard in machine translation made the Transformer an appealing architecture for other areas as well.

The transformer was designed for sequence to sequence (seq2seq) tasks, such as machine translation. It uses a special type of attention, called self-attention. In a seq2seq task, self-attention can be seen as a way to weigh which parts of the input are important for which parts of the output. Previously, in RNNs, this information was stored in the hidden state and corresponding weights. The Transformer is composed of an encoder and decoder portions, consisting of several attention and standard feed forward layers. The self-attention values are computer in the self-attention layers, which are trained using the resulting gradient.

Similar to multiple filters in convolutional networks, attention layers can have multiple heads, which should in theory allow the different heads to focus on different tasks. Similar to LSTMs, the final output of the attention layers is followed by a fully

connected layer and softmax normalization.

Non-recurrent neural network architecture based on self-attention for HAR was proposed in [65] and the results show superior performance over previously published models in the field. Although attention mechanisms and transformers were not thoroughly investigated in HAR, their simplicity and performance makes them a very viable option. One of the strengths of the Transformer over recurrent networks is the lack of recurrence, which can lead to massive increase in the speed of training and inference.

1.3 Sensor selection and placement

When designing an infrastructure for HAR or HGR, one of the important model agnostic choices we need to make is the type and placement of sensors to use. In [9] a comprehensive overview of the available sensor palette is presented (Table 1.1).

Inertial sensors	Accelerometer
	Gyroscope
	Magnetometer
Physical health sensors	Electrocardiogram (ECG)
	Skin temperature
	Heart rate (HR)
	Electroencephalograph (EEG)
	Electromyogram (EMG)
Environmental sensors	Temperature
	Humidity
	Light
	Barometer
Power sensors	Flex
	Stretch
Other sensors	Camera
	Microphone
	GPS and position

Table 1.1: Various sensor types used in HAR, adapted from [9].

The most diverse set of sensors was used during the acquisition of the OPPORTUNITY dataset[66][67] - inertial measurement units consisting of accelerometer, gyroscopes and magnetometers, 3D localization, rate of turn and various other switch sensors were used. The previously conducted surveys suggest that the use of IMUs and in particular accelerometers is receiving the largest popularity among HAR researchers, which is further supported by the fact that open datasets contain mostly accelerometer

data. This may be caused by the low entry cost and high availability of this kind of sensors and their ubiquitous presence in smartphones.

In [68], a low cost wearable glove was created using bend sensors, serving as one of the alternatives for IMUs. Intuitively, IMUs make an easier use case for HAR, while bend sensors may be more appropriate for HGR. Both of these sensor types were combined in GestGlove [69], where a specialized glove was constructed. Google's project Soli, utilizing a high frequency solid short range radar is another type of a very specialized sensor that was attempted to be used for HAR [54]. Nevertheless, for our case of attempting to build a low cost, lightweight reproducible prototype and acquiring a competitive dataset we chose to use IMUs.

Advances in compactness of the mentioned sensors enable the pervasive usage of several sensors. Data from several sensors can then be fused in various ways as examined in [42][70]. The placement of multiple sensors could in theory lead to a linear information gain with each added sensor. In contrast, an intuitive approach suggests that given the real world precision of measurements, multiple sensors placed next to each other will not provide sufficient information gain and will rather make the prototype unnecessarily cluttered.

Multiple locations were used for mounting the sensors in HAR and their optimal placement is analyzed in [10] and [71]. Arms, hips, thighs, feet, waist, back, chest, trunk, ankles, wrists or ears are some of the possible sensor locations. The published results suggest that for a single sensor setup, placement on the hip seems to be most effective in recognizing activities of daily life (walking, running, cycling, etc.). Increasing the number of sensors yields better results in [10], up to three sensors, with only marginal differences with more than two sensors, but for classifying finer grain activities, the differences may be of greater significance.

A group of 16 different activities, from the Otago Exercise Programme such as walking, backward walking, tandem walking, knee extension, leg stance and several others were classified in [52]. A set of five sensors on both feet, shins and in the lumbar area was used. Classification power of various subsets of sensor data was examined and in contrast to previous results, individual usage of the sensor in lumbar area led to lowest single sensor accuracy. The usage of multiple sensors has shown increased performance as well. This leads to conclusion that the optimal placement for IMUs can highly depend on the type of classified activities.

To the best of our knowledge, our study is the first to investigate the potential advantage of using multiple sensors placed over the hand (in the case of HGR) over using only one attached or hand held sensor.

Chapter 2

Novel dataset acquisition

To demonstrate the discriminative potential and gesture space size in a multi-modal environment, we built a custom hardware prototype called WaveGlove. We further designed a new gesture vocabulary specifically tailored for this multi-sensor setup. The dataset we recorded is competitive in both quality and size with other available state-of-the-art datasets.

For further comparability, we recorded a secondary smaller dataset using the same gesture vocabulary as the uWave dataset[15], which is used in [13] as well as [27], and an arguably similar set of gestures is used in [72], [73], [20], [38] and [19]. The main difference between our and the previously published dataset is the **usage of five sensors instead of a single one**, which creates an opportunity for an ablation study.

In general, ablation study refers to examining the effects of removing certain parts of a model or an algorithm to see how it affects the performance. The goal of such ablation study is to identify what are the key parts needed in order to achieve the reported results and also to strip down the unnecessary model complexity - from two models with the same performance, the simpler is the preferable. In our case, we will refer to an ablation study in the context of our multi-sensor WaveGlove and the datasets acquired using it. We would like to examine the added value of multiple sensors (possibly each of them) and come to a conclusion to what point does adding more sensors improve the model's discriminative potential. The ablation study is performed in Section 4.4.

Furthermore, we believe WaveGlove to be the first publicly available dataset of its kind. To the best of our knowledge, no prior work presents an Inertial Measurement Unit(IMU) based hand gesture dataset of comparable size, which uses inputs from multiple sensors. Previous surveys and overviews support this claim as well[4].

In the following sections we describe our hardware prototype and present our gesture vocabulary. Next, we provide an overview of our comprehensive recording framework, and describe the details of the experiment setup.

2.1 WaveGlove prototype

The WaveGlove prototype uses five inertial measurement units(IMUs) attached to a left-hand glove – one on each finger – to record accelerometer and gyroscope data at the rate of 40 Hz. An ESP8266 microcontroller is used to read the data from the IMUs through an I^2C interface. The ESP8266 is connected to a laptop using serial interface over USB. All of the processing happens on the laptop, the role of the microcontroller is solely to request data from the IMUs (which are behind an I^2C multiplexer) and pass it on. High level overview of the prototype structure is shown in Figure 2.1.

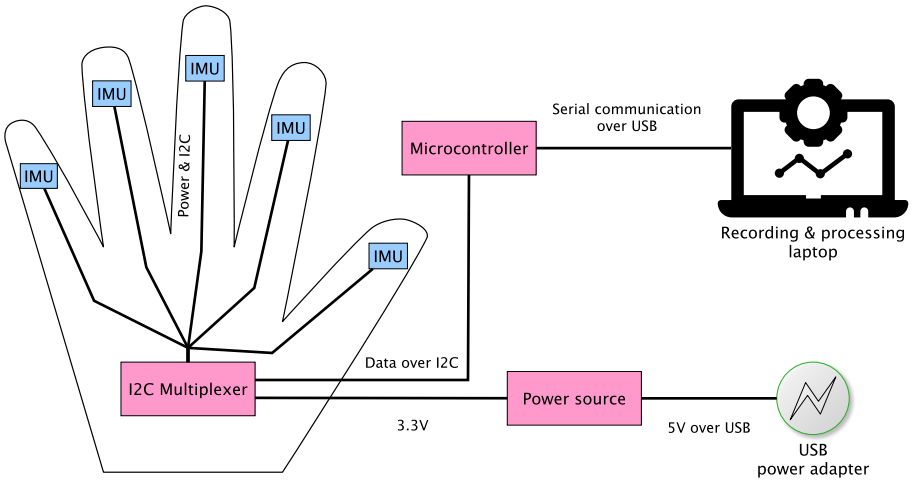


Figure 2.1: High level prototype structure.

In the sections nelow we briefly describe the motivation behind architectural choices by comparing our prototype to other alternatives. The following subsections provide detailed characteristics of the used hardware, wiring and firmware.

2.1.1 Glove-like wearables

Building a glove designed for for human activity recognition was already attempted in previous work [68][69][2]. Unfortunately the used components were either expensive, unavailable on the market, completely undocumented or not IMU based.

A few companies already provide commercially available smart wearable gloves for gesture recognition such as [74] and [75], or supportive assistance for the physically impaired [76]. Our research has led us that there is no affordable publicly available wearable solution for hand gesture recognition. This situation motivated one of our goals: to build a prototype, whose price is as low as possible, while still serving as a tool to capture a comparable dataset.

The use of commonly available components enables easier reproduction and by sticking to a low build price we try to simulate the scenario of a future mass production

of a similar device. We understand that in the case of a mass production, custom printed circuit boards would likely be used, along with machine soldered and glued connections. However, the inertial measurement unit we used is commonly found in a wide range of applications, so we assumed it to be a good fit for our case as well. Custom PCBs would likely also reduce the spatial requirements and the overall cost of the device.

In the WaveGlove prototype, all of the communication is wired. But a transition to a wireless model is not a huge technical challenge with today's technology, as we see from the similar commercially available products and case studies. Both energy usage and latency requirements are acceptable, given the batteries and wireless modules available on the market.

2.1.2 Hardware properties

The hardware structure of our prototype is described in Figure A.1. It consists of three main sections:

1. the components which are placed directly on the glove
2. the components which provide the sensors with power
3. the micro-controller used for reading the data and passing it to a recording device

We decided to use a separate power source for the sensors, because during the first tests we were not able to pull enough current from the ESP8266 to power all of them. This is in line with the specifications, as the maximum current from a GPIO pin on the ESP8266 is 12mA[77], while the current draw of an MPU6050 is more than 3.5mA[78] in our mode of use. Using the low power mode of the MPU6050 was not tested.

To test the possibility of a wireless usage, a TP4056 battery charger[79] with a micro USB connector was used along with a 3.7V 500mAh LiPo battery. The battery was able to power the whole prototype (with the ESP8266 rewired to be powered by it) for more than an hour. In further usage however, it was not used, so as to provide more stable operating conditions. Instead, the sensors were powered through the TP4056's USB input directly.

The sensors and the multiplexer can operate at 3.3V[78][80] and the XL6009 converter has minimal input operating voltage of 5V[81]. Therefore a combination of a step-up and a step-down component is used to achieve a stable voltage.

The MPU6050 sensors support both SPI and I^2C protocols for communication. Based on our experiments however, the SPI (which has lower communication overhead) was not working properly on boards from three different sellers. Due to the sensors having only two configurable I^2C addresses (based on the input on pin AD0), we used the TCA9548A I^2C multiplexer.

Individual components were chosen based on availability and price. For a component overview see Table 2.1.

Component usage	**Component name**	**Datasheet**	**Price**
Microcontroller	ESP8266	[77]	2.19 €
Charger	TP4056	[79]	0.27 €
3.7V, 500mAh Battery	JH752540P	[82]	1.20 €
Switch button	-	-	0.07 €
Step up module	SS34	[83]	0.82 €
Step down module	XL6009	[81]	0.86 €
I^2C Multiplexer	TCA9548A	[80]	0.69 €
IMU	MPU6050	[78]	5 X 0.53 €
Glove	Men's silk glove	-	4.69 €
		Total cost:	13.44 €

Table 2.1: Overview of the used components. Price estimation is valid as of May 2020 and reflects prices for low quantity, personal use offers from major online retailers.

The final prototype has about 70 cm long cable between the glove with the sensors and the rest of the system. This length was found to be sufficient in order to perform gestures naturally, without limitations. The power supply components along with the microcontroller were glued onto the walls of an enclosed carton box for protection. The carton box has therefore three connections: to the glove, to a processing device (a laptop) and to a USB power adapter. The sensors were mounted on the upper part of middle phalanges of the four fingers and on the distal phalanx of the thumb. Both the IMUs and the multiplexer are glued to a hook-and-loop fastener of which the other part is glued using a textile glue to the glove itself. This allows detaching the sensors for easier manipulation, cleaning and possibly replacement of the glove. Worn prototype is shown in Figure 2.2, its casing in Figure A.2 and animation of performing a movement in Figure 2.3.

2.1.3 Firmware

We tested three different microcontrollers in the setup, and based on the requirements and tests described below we finally used the ESP8266, as it was the most stable and performant variant. The PlatformIO build system[84] was used to create the firmware for the ESP8266.

After startup, the microcontroller searches it's ports for a valid I^2C connection to the multiplexer. Consequently, the range for the sensors is set and a short calibration is done for each of the five sensors For communication with the recording device (typically

Figure 2.2: Five sensors are attached to the middle phalanges on the fingers and the multiplexer is located in the middle of the back of the hand. All connections are covered with glue for protection and sturdiness.

Figure 2.3: Cables leading from the multiplexer to sensors on the fingers are long enough to be non-obtrusive. Performing movements while wearing the glove does not present any major difficulties.

a laptop) serial communication via the USB connection is used.

By examining various baud rates from 9600 to 1843200 (10 bauds are needed to transfer 1 byte of data on our platform), we found 460800 to be the highest stable baud rate that the microcontroller was able to maintain. With higher baud rates, the error rate was unreasonably high and the ESP8266 self-restarted after a few seconds (up to a minute).

Accelerometer range of $\pm 4g$ and gyroscope range of $\pm 1000°/s$ was used. Based on a few experiments, these were the highest precision settings, for which the values did not get clipped when performing the gestures. Inspired by other MPU6050 projects, we calibrate the sensors by taking 100 subsequent readings from each of the sensors and averaging them. This value is then used as an offset for each of the subsequent readings.

When all of the five sensors are calibrated, the main sensor reading loop takes place. In each cycle each of the five sensors reads six values, each consisting of two bytes, interpreted as a signed integer. We use a specific combination of four bytes as a marker for end of the cycle, leading to a total cycle length of $12 * 5 + 4 = 64$ bytes.

In theory, the combination of bytes used as end marker could also appear in the readings themselves, leading to invalid segmentation of the streamed data. Experiments have shown that this is very unlikely and in combination with a proper software implementation can so that the error rate is kept at minimum.

Benchmarks of the reading cycle have shown it to take between 7 ms and 12 ms, which could lead to frequency around 80 Hz. However, consequent tests have shown that not limiting the calculation with delays again leads to instability in the microcontroller behaviour. We tested reading frequency of multiples of 10 and found 40 Hz to be the fastest achievable stable reading frequency in our setup. Frequency stabilization is implemented by a short sleep. Each reading cycle of length t ms is followed by a sleep $25 - \text{t}$ ms long.

Other studies contain readings from sensors ranging from 30 Hz up to 1000 Hz or even more in the case of non-IMU sensors[66]. Therefore we believe that the chosen frequency of 40 Hz can lead to competitive dataset quality. Increasing the frequency could likely be achieved in various ways:

- Use a different microcontroller, which supports higher clock speed and is stable at higher baud rate.
- Switch the protocol from I^2C to lower overhead SPI, or use multiple I^2C addresses instead of a multiplexer.
- Explore the possibility of reading the data from sensors in parallel.

As our goal is to simulate a realistic scenario using low cost and widely available components and the achieved rate is sufficient for state of the art, we decided not to pursue increasing the frequency further. Table 2.2 summarizes the configuration parameters used.

Parameter	Value
Reading frequency	40 Hz
Frequency variations	±2 Hz
Acceleration range	$\pm 4g$
Angular velocity range	$\pm 1000°/s$
Single reading size	2 B
Read cycle size	2 B
Serial baud rate	460800

Table 2.2: Hardware configuration

2.2 Gesture vocabulary

In the area of human activity recognition (HAR), previous work focuses mostly on full body activity recognition [7] (e.g. walking, running, sitting, jumping), while the hand gesture recognition (HGR) sub-area of HAR is more often approached using optical input (such as cameras). Recent field reviews[4] suggest that hand gesture recognition is mostly explored using only a single input sensor as in [15] and only a few sources examine the potential of a multi sensor hand gesture recognition[69][68][2].

Publicly available datasets are also mostly focused on full body HAR rather than HGR. To the best of our knowledge, there is no significantly large, publicly available, well documented, multi sensor HGR dataset. The closest work that shares our goal is [2] which has shown very promising results in continuous recognition of the gestures in the french sign language alphabet.

In order for our work to create a novel dataset which embraces the multi sensor potential, while still being comparable to previous work, we decided to record two datasets. The first one is aimed at comparability with previous work and providing the opportunity of an ablation study and the second dataset tries to explore all the possibilities of intuitive but composite movements of the five fingers.

2.2.1 Dataset WaveGlove-single

For the first (smaller) dataset, we used a vocabulary of eight gestures, which is widely used in previous work[15][13] and [27]. It is comprised of single-point movements,

because they were originally designed for a single sensor device. We record over a thousand of samples comprising of these eight gestures and a null class. The null class contains samples recorded during breaks between performing individual gestures. Due to the nature of the movements, we call these gestures **single gestures** and the corresponding recorded dataset **WaveGlove-single**. Each of these eight gestures is described only using a single arrow for the movement direction as shown in Figure 2.4.

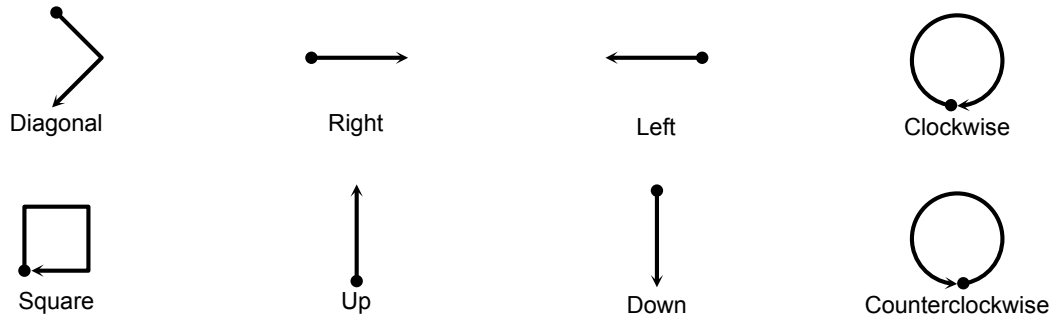


Figure 2.4: Gestures in the WaveGlove-single dataset. All of them represent only simple, single-point movements.

2.2.2 Dataset WaveGlove-multi

One of our main contributions is the second (larger) dataset, consisting of ten different gestures and a null class. Due to its size being more than ten thousand instances, we believe that it is the largest publicly available dataset focusing on hand gestures. Note that there exist datasets focusing on HAR instead of HGR which easily contain much more instances, due to the segmentability of long term recordings. We call gestures from this vocabulary the **multi gestures** and the dataset **WaveGlove-multi**.

All of the ten gestures are described using a GIF animation with a virtual hand performing the motion. This animation is seen by the test subject during the testing session (more information on the experiment setup is provided in Subsection 2.4). The original animation files are distributed with the dataset and in Figures A.3 and A.4 we show the ten gestures using static images (the arrows are used for clarification and are not present during experiments).

Hand swipe left and *Hand swipe right* gestures represent simple, whole hand movements, where each finger exhibits very similar movements. *Pinch closer* and *Pinch away* are named after the typical zooming actions on a touch screen - pulling the fingers together zooms out (away) and spreading them zooms in (closer). *Thumb double tap* is a quick and simple thumb-only gesture. The bumps of the thumb into the other four fingers do not present large positional changes, but introduce temporary acceleration spikes for all fingers. *Grab* and *Ungrab* involve flexing all of the five fingers and

Page flip introduces another whole hand movements, this time with a sharp change of orientation. *Peace* and *Metal* gestures focus on utilizing only a specific subset of all fingers in an asymmetrical situation.

Six of the ten gestures can be grouped into three pairs of opposite actions (the swipes, pinches and grabs), making it a classification requirement to distinguish between complementary movements. The complete set of gestures contains both whole hand movements, finger-agnostic movements as well as movements with different behaviours for each of the fingers. The chosen gestures could also find their use in augmented reality device control - grabs and ungrabs with hand movements could move around some objects, pinches could be used for zooming and swipes for some switch requests. More information on both datasets can be found in Chapter 3.

2.3 Recording framework

To facilitate the recording of the dataset, we created a GUI application using the PyQt[85] libraries. The created framework provides the following functionality:

- Connect to a WaveGlove prototype (or a device adhering to the same protocol) on a chosen USB port through Serial communication. Automatic reconnection further enhances the ease of use.
- Display connection status and periodic statistics about the readings.
- Real time monitoring of the current sensor readings.
- Choose and start a recording session. The sessions are defined using specifically structured YAML files and consist of individual slides. More details on what a session looks like can be found in section 2.4.
- Browse previously recorded sessions, preview stored metadata, information about recorded gestures and plot the recorded data.

Additionally, we implemented features which are not necessary for the final dataset acquisition, but are useful for testing purposes. These include recording a single gesture instance, plotting and management of previously recorded gestures, and creation of a virtual serial port with simulated data for debugging purposes.

Rich functionality, high configurability and framework portability are factors which we believe allow the created framework to be used with similar devices in further work. Figure 2.5 shows the created application.

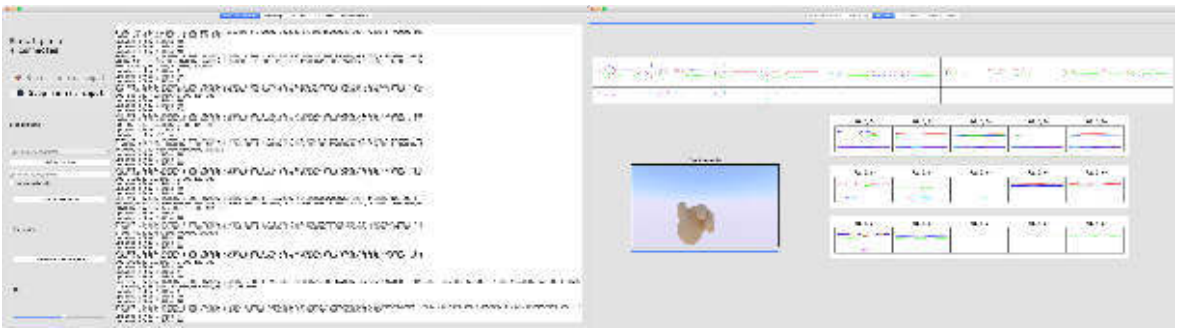


Figure 2.5: **Recording application** - After startup a serial port is selected and a connection is created. The interface is separated into tabs. On the left we see the connection tab showing us the state of the connection and all the relevant status logs. Session recording tab is on the right in the process of recording the *Thumb double tap* gesture.

2.4 Experiment setup

In this section we first describe the properties of the recording sessions we designed. Then we consider various factors that could influence the quality of the created dataset and discuss them further.

During a **recording session** a test subject performs gestures using the equipped prototype. For each gesture class, three consecutively performed instances are recorded. The order of the classes is randomly shuffled for each session and between each of the classes a mandatory 2.25 s pause is inserted, which serves as a separator. Together, these two factors make the gesture performance less dependent on the context (gestures prior to an after it) and allow approaching each gesture individually instead of performing a pre-defined routine.

The test subject is not given any specific instructions on what to do during the pause, therefore we record the sensor data during this time and use it for the null class. This way we effectively make the null class some unspecified movement, and very likely not a gesture. During the recording, the pause time was usually used by the subjects to rest their hand or make a random stretch.

When recording the three consecutive instances for each class, the test subject annotates the beginning and end of each instance manually, directly during the recording. For each class, the subject is presented with a "slide" containing the following three components:

1. animation or an image and possibly the name of the class.
2. live signal data from the sensors.
3. overview of the already recorded instances.

The live signal data, and overview of the recorded instances serve only for a basic sanity check for the subject (e.g. to prevent recording when the sensor connection is failing, or to prevent mistakenly recording zero or more than one gesture).

The subject is instructed to first study the gesture he or she is about to perform. and start the process at their own convenience. Recording of a gesture instance starts by pressing the space key and ends with its release. The pause between the three instances in a single class is determined by the pace of the subject, no limit is enforced. The minimal instructions the subject is given do not specifically set an expectation for gesture orientation or speed of performance. Only in the WaveGlove-multi dataset a small progressbar is shown under the animation to allow better understanding of the movement (experiments have shown that subjects had otherwise troubles discriminating between consecutive animation cycles).

It is not unlikely for the subject to make a mistake - forget to press or release the space key in time, perform the wrong gesture, be interrupted by an external factor, etc. In this case, the mouse cursor can be used to re-select the malformed instance and it can be recorded again. This helps to minimize problematic samples, which is vital for the dataset. Slides from a tutorial session can be seen in in Figure A.5 and other examples from recording a session are in Figure A.6.

2.4.1 Qualitative dataset factors

There are several factors affecting the quality of a dataset in similar experiments [86] and in our experiment we considered several of them. An outline of considered factors is presented in Table 2.3.

We acknowledge that some qualitative measures failed to be taken and their implementation would improve the dataset usability. The number of test subjects is very limited (two people - one male, one female), which was an unfortunate consequence of the hygienic requirements of the worldwide pandemic situation in spring 2020. However, the SKODA Mini Checkpoint dataset, which is widely used also contains recordings of a single test subject[87][88], and our class-subject balance is far superior to that of the PAMAP2 Dataset[89]. This leads us to believe this limitation is not a fundamental issue. The recording frequency of our dataset is 40 Hz, which lies in the standard range of public datasets. With the proper hardware and expertise this could be increased two or three-fold without major change in characteristics of the system and with only minor impact on the overall cost of the prototype. The effect on performance of acquiring data with higher frequency remains questionable.

Category	Factor	Resolution
Hardware	Stable sensor frequency	Hardware limitations were explored and stabilization occured up to 2Hz precision.
	Reproducibility	Prototype was built from generally available components and extensively documented.
Quantitative	Size	Instance counts of 1000 for WaveGlove-single and 10000 for WaveGlove-multi are competitive with state of the art.
	Longer time frame	Session were recorded over the course of 16 days which is considered appropriate.
	Null class	Almost one fourth of the recorded data represents the null class, leading to vitally imbalanced dataset.
Design	Relevant vocabulary	WaveGlove-multi vocabulary was tailored to explore the limits of five sensors placed on fingers.
	Ease of use	Custom tutorial session was designed along with a user friendly recording framework.
	Mistake removal	Subjects are allowed to re-record an instance in case they make a mistake (e.g. double press).
	Reproducibility	Subjects received uniform instructions and the framework along with all relevant materials is published, allowing reproduction experiments.
Qualitative	Order of performance	In each session target classes are shuffled, effectively preventing mundane execution of a learned pattern.
	Independence of instances	Forced break is inserted between recording each of the classes, which is long enough for the subject to relax temporarily.
	Realistic null class	Recording continues during the forced break and is used for the null class. This information is not explicitly disclosed to the subjects, nor they have specific instructions on what to do during the break.

Table 2.3: An overview of various qualitative dataset factors considered.

Chapter 3

Datasets for experiments

For evaluation of the HAR and HGR methods we decided to use our acquired datasets as well as several publicly available ones. This way our methods can be validated on an extensive amount of data from various sources, and the relevance of our newly created dataset will be asssessed through direct comparison.

In addition to the newly acquired WaveGlove datasets we preprocessed five widely used datasets. As we note in Section 3.1, the method used for generating samples from the raw signal and other preprocessing factors highly impact the classification results. Therefore we also include a set of six datasets preprocessed in [8] using two of the four methods they suggest. An overview of the considered datasets used is in Table 3.1.

Datasets with implicit sample generation or non-overlapping samples	Datasets with samples as generated in [8]
WaveGlove-single	
WaveGlove-multi	USC-HAD
uWave	UTD-MHAD1
OPPORTUNITY	UTD-MHAD2
PAMAP2	WHARF
SKODA	WISDM
MHEALTH	MHEALTH

Table 3.1: Overview of the dataset sampling. The MHEALTH dataset is used both using non-overlapping sampling as well as using two other sampling methods.

Comparison of the WaveGlove-single and the uWave datasets is of particular interest, because their gesture vocabulary is the same. However our WaveGlove-single uses five times the amount of sensors and ten times the amount of channels. The impact of the significant difference in the number of channels is investigated in an ablation study in Section 4.4.

The WaveGlove, uWave and UTD-MHAD1 are segmented HGR datasets, while the other investigated datasets are continuously recorded HAR datasets, where the duration of each of the actions is higher.

As part of our work, we publish all of the datasets in a uniform, preprocessed form that can be used for further research (all required permissions were requested and citations included). Datasets are stored in the HDF5[90] file format, which is highly optimized for heterogenous data, I/O performance, allows easy sharing and has cross platform support. Metadata is also saved with the data itself, which further streamlines processing pipelines. The H5Py library[91] was used for data-related operations.

In the following sections we discuss the methods used for data preprocessing and describe the setup of experiments for both types of included datasets and present summary statistics.

3.1 Preprocessing datasets

The available HAR and HGR datasets differ most fundamentally in the way they are recorded. In HAR, the goal tends to be the classification of a long-term state lasting from several seconds to minutes and it usually builds on a full body posture: walking, sitting, lying or computer work. In HGR, the goal is to classify a short-term gesture, local to the hand (or hands) which typically lasts up to two or three seconds. HAR could be used for all-day tracking of activities, while HGR focuses on short actions that can serve as triggers for the environment.

Due to the fact that HGR tries to classify individual occurrences of short gesture and HAR is rather focused on a state-like classification, the recorded datasets differ slightly based on their target use case. Datasets from HGR are usually presegmented: during or after the recording, the individual gesture occurrences are separated and are relatively short. The resulting data therefore consists of a large amount of shorter instances. Datasets from HAR usually consist of fewer longer instances from several minutes long recordings. Sub-sequences of these long continuous recordings are then labeled accordingly. The character of the samples between these labelled sub-sequences depends on the experiment setup. It can either be used as a null class, if the experiment was designed to be a single, longer run, where the subject really "does nothing" in between specific activities, or they could be discarded if the time between specific activities was used for calibration, sensor adjustments, or just not tracked (e.g. the subject had the sensors equipped during a transition between the places required to perform the activities).

The HGR datasets usually come split into separate gestures, but the HAR datasets need to be split into smaller samples which will be used for classification. The used

sample generation and train-test split process can have significant impact on the final results[8]. In [8] four approaches are described - *Semi-Non-Overlapping-Window*, *Full-Non-Overlapping-Window*, *Leave-One-Subject-Out* and *Leave-One-Trial-Out* as a new method. In all four approaches the data is segmented into same-length temporal windows, which receive either the last label from the sequence, or the label that appeared most often during the window.

Semi-Non-Overlapping-Window creates a sequence of windows, where two consecutive windows overlap by 50% - this approach is used most commonly in previous work. A notable drawback of this process which can lead to skewed results is that improper validation metric can cause high bias. If the windows i and $i + 1$ appear in different folds of cross validation (or other protocol), then the 50% overlap of the windows causes the overlap to be present in both training and testing sets at the same time, biasing the results. This flaw was not considered in previous work, therefore the reported results do not reflect real performance[8]. Other work failed to mention the segmentation technique completely, effectively preventing any attempts at reproduction.

Full-Non-Overlapping-Window refers to extracting consecutive windows without an overlap. Even though this method does not experience the bias described previously, it has a slight disadvantage in generating fewer samples. In [8] the *Leave-One-Trial-Out* method was introduced to prevent both of these drawbacks and *Leave-One-Subject-Out* is the approach typically used in previous work.

As the dataset sizes are sufficient for our experiments in the *Full-Non-Overlapping-Window*, we do not search for a way to increase the number of samples. We would, however, like to improve the length distribution of the generated samples. The application of the aforementioned methods leads to constant length sampling and slight divergence from a real-world scenario. Therefore we propose the *Non-overlapping semi-uniform length windowing* approach and use it on the relevant datasets.

3.1.1 Non-overlapping semi-uniform length windowing

In order to achieve slight differences in the window lengths and utilize the majority of the data, we propose the following method. A single global parameter is chosen for each dataset – the minimal segment length l_{min}. We call each continuous sequence labelled with a single class a *run* and its length l_{run}. For each run, the number of minimal segments $c_{segment}$ that can fit into it is calculated. If the *run* is shorter than a minimal segment, then the whole *run* is used as a single sample in the output set. Otherwise, we calculate the largest possible length of which $c_{segment}$ segments can fit into the *run*: $l_{final} = l_{run} \div c_{segment}$. At the end, $c_{segment}$ samples, each of length l_{final} are added to the output set. Note that the remaining $l_{run} - c_{segment} \cdot l_{final}$ readings are ignored. The amount that gets ignored this way is negligible. A diagram visualizing

the process can be seen in in Figure 3.1.

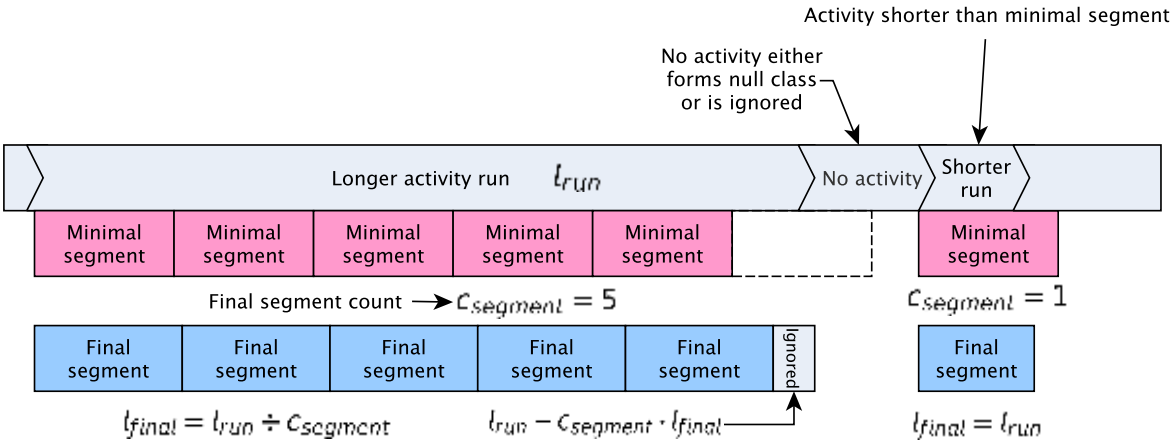


Figure 3.1: Non-overlapping semi-uniform length windowing, applied to two runs. The final segments are added as samples to the output set.

As the output, we create a set of samples with the following properties. Runs shorter that the l_{min} parameter are not affected and included as separate samples. Other runs are split into smaller samples and their length is identical in a single run, but may differ between runs. The upper bound on l_{final} is $2l_{min} - 1$ in the rare case when the length of the run is $2l_{min} - 1$ (if the run has length of at least $2l_{min}$, it is split into more than one sample). This windowing technique is used in the preprocessing of various datasets as noted below.

3.2 Dataset overview

We chose the uWave dataset as a dataset with equivalent gesture vocabulary and one of the few datasets that focus on HGR. We also acquired the dataset from [2] but due to the different preprocessing approach, ultimately it was not used (only the filtered pitch and roll calculated values for each of the five sensors were available).

Based on existing field surveys [86][6] [4], we chose the OPPORTUNITY, PAMAP2, Skoda and MHEALTH datasets as one of the most prominent examples of benchmark datasets in the field of HAR.

3.2.1 WaveGlove-single and WaveGlove-multi datasets

Both WaveGlove datasets were acquired as described in 2. A custom glove was created and a single IMU (with accelerometer and gyroscope) was mounted on the middle phalanx of each finger (distal phalanx of thumb). A vocabulary of eight (WaveGlove-single) and ten (WaveGlove-multi) gestures was chosen and null class samples are

included. Further information about the experiment setup is in 2.4 and the gesture vocabulary in 2.2.

3.2.2 uWave dataset

Containing over 4000 samples, the uWave dataset[15] was acquired for low training sample personalized gesture recognition. The eight gestures were identified by a Nokia researchers to be the ones preferred for mobile interaction. Samples were recorded on a Wii remote controller, which has a built-in three-axis ADXL330 accelerometer[92] with a range of $\pm 3g$ and operating frequency of 100 Hz. Eight right handed participants (7 male, 1 female) in their 20s or early 30s participated in data collection.

Gestures were collected during seven days within a period of about three weeks. On each day, the participant repeats each of the gestures in the vocabulary ten times. The dataset and the associated work further highlight that inter-user and inter-day variations in gesture pose a significant challenge for recognition and not accounting for this may lead to overly optimistic results. Gesture vocabulary has been previously shown in 2.4.

3.2.3 OPPORTUNITY dataset

Probably the most prominently used dataset in HAR is the OPPORTUNITY dataset [66][67]. The original dataset contains readings from motion sensors recorded while the subject executed typical daily activities in a room simulating a studio flat. A large variety of sensors has been used in the experiment - 7 IMUs, 12 three-axis accelerometers, 4 3D localization sensors, 12 sensors attached to objects measuring acceleration and rate of turn, 13 ambient switch sensors and 8 three-axis accelerometers attached to appliances and furniture.

A distinguishing feature of this dataset are its multi-level annotations. Four separate annotation tracks are provided: four modes of locomotion (sit, stand, lie walk), rich set of low-level actions relating actions with objects, 17 mid-level gesture classes (open/close door, drink, ...) and 5 high-level activity classes. Four people took part in the creation of this dataset, and six runs per each of them were recorded. Five of the runs are naturally executed daily activities and the sixth run is a drill run, where a scripted sequence of activities is performed. The natural runs build a set of temporarily unfolding situation during a typical kitchen routine: relaxing, preparing and drinking coffee or preparing and eating a sandwich. The data contains "NaN" values due to connection problems with wireless sensors. The overall sample rate was 30 Hz, but the data or the annotations may contain slight jitter of the order of 100 ms due to the complexity of the system.

A subset of this dataset was used for the OPPORTUNITY Activity Recognition Challenge held in 2011. Prior work follows the recommendation of the authors to only use the data from on-body sensors for HAR and we do so in our work as well. This includes the seven IMUs (five of which are placed over the upper half of the body and two of them are located in the shoes), twelve three-axis accelerometers and the four localization sensors.

For target classes, we use the middle level gesture annotation track, again following previous work and recommendation of the authors. Unlabeled segments of the data are assigned to the null class, yielding a total of 18 target classes. The middle level activities include opening and closing doors, the fridge, the dishwasher, various drawers and cleaning the table, drinking and toggling switches. Samples are collected using the proposed non-overlapping semi-uniform length windowing with l_{min} corresponding to 1 s temporal window, creating a highly imbalanced dataset. The null class forms approximately three quarters of the data, so the usage of non-uniform sampling techniques may be necessary for achieving optimal performance. Information about sensor placement and experiment setup is shown in Figure B.1.

3.2.4 PAMAP2 dataset

In the PAMAP2 dataset[89] various physical activities are recorded. The sensors consist of three IMUs with sampling frequency of 100 Hz and a heart rate monitor with sampling frequency of 9 Hz. The three IMUs are placed on the chest and on the wrist and ankle on the dominant side of each subject. In total 9 subjects participated in the experiment (8 male, 1 female) aged 27.22 ± 3.31 years and with a BMI $25.11 \pm 2.62 kgm^{-2}$.

Each of the subjects followed a protocol containing 12 different activities, and a list of optional activities to perform was also suggested to the subjects. In total 18 different activities were performed. Parts of the data labelled as the null class are ignored as recommended by the authors, as this time covers transient activities between performances, e.g. going from one location to the next activity's location. The activities are whole body state and focused on various physical action in one's daily life - lying, sitting, watching TV, car driving, ironing, rope jumping, etc. Each of the activities took $1 - 3$ min. Parts of the data contain "NaN" values due to dropping connections and the lower frequency heart rate monitor contains "NaN" values at time points it did not provide a reading. Therefore some further processing of the heart rate channel is required.

For our analysis, we used the acceleration data on the scale $\pm 16g$ (they were also provided in higher resolution, but those reading got saturated during certain activities), gyroscope data, magnetometer data and the data from the heart rate sensor - providing

a total of 28 channels. Temperature data was not processed for our evaluation.

Activities in the dataset are only slightly time-imbalanced but due to the inclusion of the optional activities not all activities were performed by all subjects, and even activities performed by only one of the subjects are present. The activity distribution is shown in Table B.1. Samples are collected using the proposed non-overlapping semi-uniform length windowing with l_{min} corresponding to 1 s temporal window.

3.2.5 Skoda Mini Checkpoint dataset

Ten manipulative gestures from a car maintenance scenario are captured in the Skoda Mini Checkpoint dataset [87][88]. These are a subset of 46 activities performed in a car factory in one of the quality control checkpoints. With a sampling rate of 98 Hz, 10 three-axis accelerometers were placed both on left and right arms of a single subject. Each of the ten activities was recorded 70 times. Samples are collected using the proposed non-overlapping semi-uniform length windowing with l_{min} corresponding to 1 s temporal window. Sensor placement and gesture descriptions are in Figure B.2.

3.2.6 MHEALTH dataset

Another widely used dataset containing various physical activities and exercises is the MHEALTH dataset[93][94]. The collected dataset was acquired from ten volunteers of diverse profile when performing twelve physical activities (e.g. standing, walking, climbing stairs, cycling). A three-axis accelerometer was placed on the chest along with a 2-lead ECG sensor. Another IMUs recording acceleration, angular velocity and magnetic field orientation were placed on subjects right wrist and left ankle (attached using elastic straps), creating a total of 23 channels. The various activities involved focus on different body parts, resulting in differences between the states of the sensors. Sessions were also recorded using a video camera and the subjects were given no specific instructions with the expectations to try their best.

All sensors provided a stable sampling rate of 50 Hz and the unsegmented activity lengths are about 1 min. In our experiments, we used the data preprocessed in the work of [8], using the *Full-Non-Overlapping-Window* technique, allowing direct performance comparison with their work. Null class readings were not used and the segmented activities have constant length

3.2.7 Dataset summary

Ultimately we collected a set of seven datasets, of which two were recorded on our prototype, three are focused on gestures and six use more than a single sensor. Class distribution of the datasets is shown in Figure B.3 – datasets containing null class

tend to be less balanced. Figure B.4 shows the length distributions of the samples. The WaveGlove datasets and the uWave dataset provide higher variation in sample length, due to their record-time segmentation character. OPPORTUNITY, PAMAP2, and Skoda datasets show lower variance in sample length and were segmented by the method we proposed, while the MHEALTH dataset has constant length samples. Table 3.2 summarizes qualitative properties of the datasets and Figure 3.2 shows the quantitative properties.

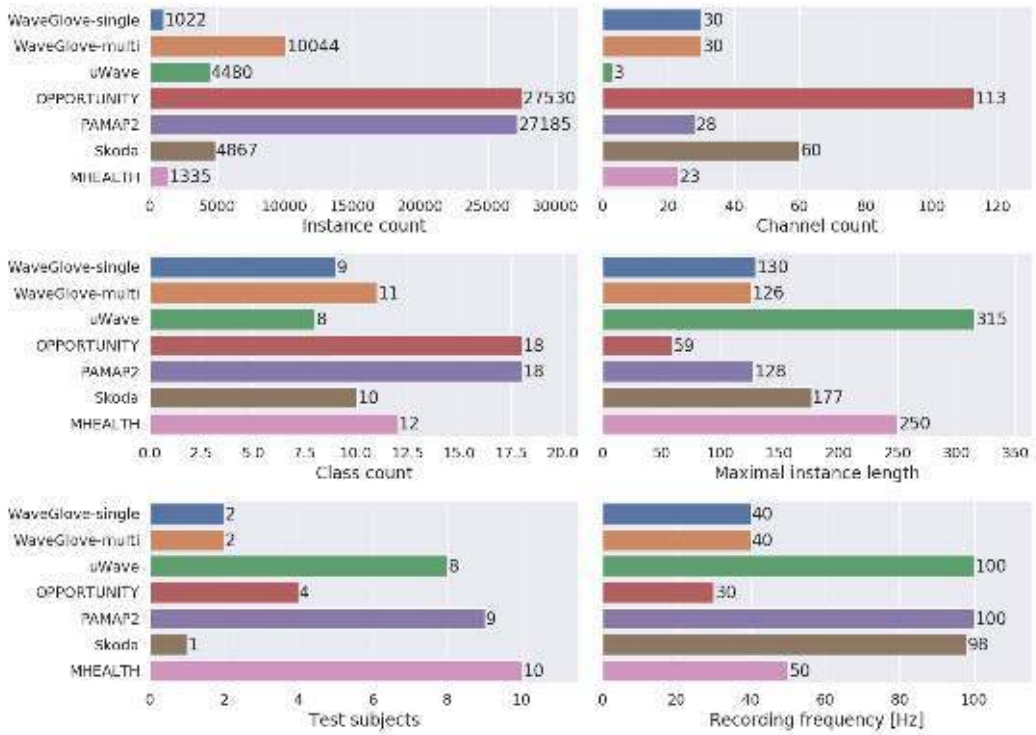


Figure 3.2: Comparison of quantitative dataset properties.

	WaveGlove-single	**WaveGlove-multi**	**uWave** [15]	**OPPORTUNITY** [66]	**PAMAP2** [89]	**Skoda** [87]	**MHEALTH** [93]
Activity type	Hand	Hand	Hand	Daily	Physical	Work	Physical
Null class	Yes	Yes	No	Yes	No	No	No
Segmentation	None	None	None	NOSU	NOSU	NOSU	FNOW
Year	2020	2020	2008	2011	2012	2007	2014
Total length	-	-	-	±3 h	10 h	±3 h	±3 h
NaN values	No	No	No	Yes	Yes	No	Yes

Table 3.2: Qualitative properties of the datasets. Legend: NOSE - Non-overlapping semi-uniform, FNOW - Full-Non-Overlapping-Window.

3.3 Externally preprocessed dataset overview

To achieve the most direct and accurate comparison of model performance, various factors need to be considered. Most notable work in this area is that of [8], in which the authors publicly provide six preprocessed datasets. Not only are the input values and target labels are provided uniformly for these datasets, but the fold data is also included. Using the same exact set of folds for (cross) validation yields better model comparability. These datasets are provided in four different modes of preprocessing - *Full-Non-Overlapping-Window* (FNOW), *Semi-Non-Overlapping-Window* (SNOW), *Leave-One-Subject-Out* (LOSO) and *Leave-One-Trial-Out* (LOTO).

In the previous section, we used the MHEALTH dataset preprocessed in the *Full-Non-Overlapping-Window* mode for comparison with the datasets we preprocessed. For additional comparison we will also adopt all of these datasets preprocessed in the *Leave-One-Subject-Out* and *Leave-One-Trial-Out* modes. *Leave-One-Subject-Out* is most widely used in previous work, while *Leave-One-Trial-Out* is shown to be less biased and has more realistic results. We would like to emphasize that [8] did make these datasets along with the fold data publicly available and previous work such as [95] used them to increase result reproducibility.

Below we briefly describe these datasets. The MHEALTH dataset was already described in the previous section, so we leave it out here. Collective summary of the externally preprocessed datasets can be seen in Table 3.3.

USC-HAD

A single wired device packed into a mobile phone poach was used for data collection. The used device was wired to a miniature laptop used for the data collection. Subjects have worn the poch at their front right hip and held the laptop in one hand. Activities were performed only with high-level instructions, leading to higher user-dependent variations. Each subject performed 5 trials for each activity on different days at different indoor and outdoor locations.

In general, it took about 6 hours to finish the whole data collection for each of the 15 subjects. Data was annotated by an observer during the recording. Activity vocabulary consisted of common whole body movements, such as walking forward, walking in a circle, running, jumping, standing, riding and elevator, etc.

UTD-MHAD

The original multimodal action dataset used a combination of color camera, depth camera and a 9-axis IMU to capture a set of 27 activities. For HAR, the image data and magnetometer data are discarded leaving a total of 6 channels. Each of the 8 subjects repeated each action 4 times.

The action vocabulary can be split into 21 classes of movements performed by hand (with the accelerometer attached on the right wrist) and 6 classes of whole body movements (with the accelerometer attached on right thigh). These two types of activities are benchmarked separately under names UTD-MHAD1 and UTD-MHAD2. UTD-MHAD1 vocabulary includes actions such as arm swipes, claps, drawing x's and circles or knocking on an imaginative door. UTD-MHAD2 voacbulary consists of jogging, walking, sit to stand, stand to sit, forward lunges and squats.

WHARF

Wearable Human Activity Recognition Folder (WHARF) dataset is a public collection of labelled accelerometer recordings. After preprocessing a set of 12 classes including brushing teeth, lying, sitting, drinking, pouring water or climbing the stairs is included in the dataset.

WISDM

Six different activities were performed in controlled laboratory conditions in order to collect the data. A custom Android application was used, collecting acceleration data every 50 ms. Action vocabulary includes outdoor activities such as walking, jogging but also climbing stairs, sitting and standing. The dataset is slightly imbalanced, with 37.2% of the labels referring to walking and only 5% referring to standing.

Dataset	Frequency (Hz)	# Sensors	# Channels	# Activities	# Subjects	# Trials	# Samples	Balanced
MHEALTH [93]	50	3 (Acc, Gyro, Mag)	23	12	10	262	2555	Yes
USC-HAD [96]	100	2 (Acc, Gyro)	6	12	15	840	9824	No
UTD-MHAD1 [97]	50	2 (Acc, Gyro)	6	21	9	617	3771	Yes
UTD-MHAD2 [97]	50	2 (Acc, Gyro)	6	6	9	190	1137	Yes
WHARF [98]	32	1 (Acc)	3	12	17	884	3871	No
WISDM [46]	20	1 (Acc)	3	6	36	402	20846	No

Table 3.3: Summary of the dataset properties as shown in [8]. Reported sample counts are the result of applying the *Semi-Non-Overlapping-Window* sampling.

Chapter 4

Classification

We compare a total of nine different classification methods on the preprocessed datasets. We present two baseline non-DL methods - Average representative and Bagging Decision trees. For DL learning methods, we compare various neural network architectures - linear, convolutional, recurrent, attention-based and their combinations which were proposed in previous work. For the newly proposed methods we perform a hyperparameter searches.

First two methods were implemented using the scikit-learn library[99] and for neural networks we used the PyTorch framework[50]. Code was run on a machine with the Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz processor (6 cores, 12 threads) with a single GeForce GTX 1080 graphics card (8 GB of memory, CUDA version 10.2). For neural networks, batch sizes were set to 64 or 128, with no significant difference in performance between these two options.

Throughout the description of classification methods, we use the following notation:

B - Batch size

S - The total number of channels from all sensors

T - Temporal length of a sample (zero padded to be constant for each dataset)

C - Target class count

For evaluation we report Accuracy, Recall and Macro F1 score. These metrics can be calculated from four variables: TP (True Positive, correctly recognized as label), TN (True Negative, correctly recognized as non-label), FP (False Positive, incorrectly recognized as label) and FN (False Negative, incorrectly recognized as non-label).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Recall = \frac{TP}{TP + FN}, Precision = \frac{TP}{TP + FP}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In a multi-class scenario, there are various ways how to calculate the metrics - based on previous work we use macro averaging. In macro averaging, the metric is computed for each label separately and then the average is reported.

Various training and testing schemes can be used for evaluation and [8] shows that they have a huge impact on the resulting metrics. We report three types of results in the rest of the chapter.

First, we evaluate models on the datasets with non-overlapping samples (left column in Table 3.1) using a single random cut to split the data into 85% set and a 15% test set. Afterwards, the 85% is randomly fold k (specific for a method) times. During each fold $\frac{k-1}{k}$ of the samples are used for training and $\frac{1}{k}$ samples are used for validation (e.g. to decide when to stop training in order to prevent overfitting). From the models trained during the k folds, the best performing one (based on its F1 score) is chosen for the final evaluation on the test set. The resulting metrics are reported. This process is shown in Figure 4.1a.

For the rest of the datasets (right column in Table 3.1), we use the predefined LOSO and LOTO folds from [8]. The number of folds is either equal to the number of subjects (LOSO) or it is set to 10 (LOTO). Mirroring the evaluation used by [8][45][100], we divide the dataset using the pre-defined fold into a test set and and the rest. The rest is then randomly divided into a 85% training set and a 15% validation set. A considered model is trained using the training set and the validation set is used to stop the training. Afterwards, the trained model is evaluated against the single pre-defined fold (test set). This is consequently repeated for each of the folds and we report the average of the metrics over all folds. This process is described in Figure 4.1b.

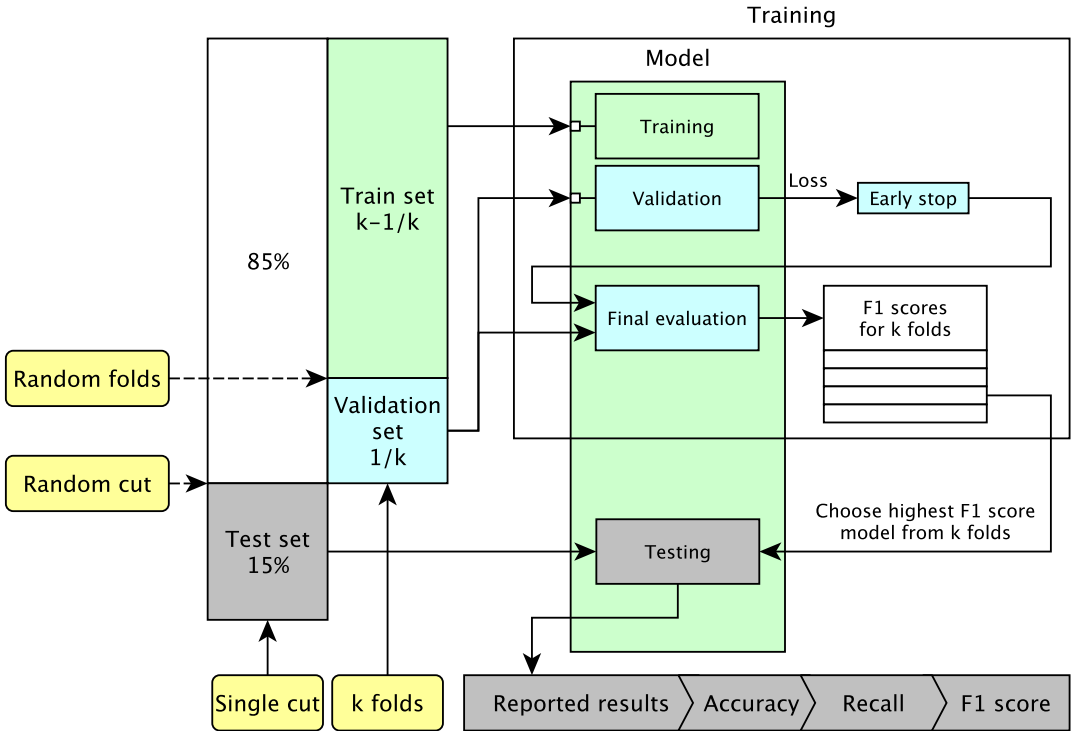
For the non-DL methods, we use a 90% training and 10% test split, with no validation set used.

In the following sections we present the non-DL method, DL methods and summarize the results of previous and proposed methods. Finally, we perform an ablation study on the WaveGlove datasets in which we investigate the impact of the number of sensors and the size of the training set size on model performance.

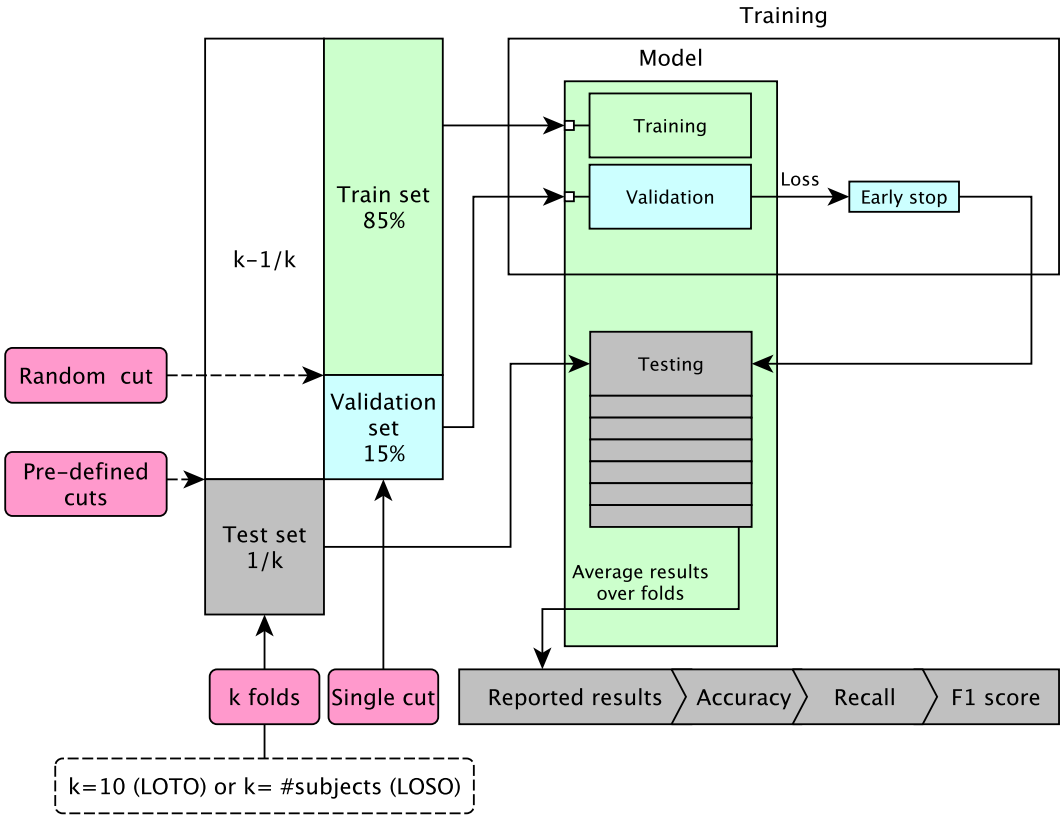
4.1 Baseline methods

To better understand the performance improvements introduced by various neural network architectures, we present two non-DL methods that we can use as a baseline.

The first method is a very naive approach, which will set a lower limit for any meaningful classification. We interpret each of the recorded samples (i.e. gesture or activity) as a point in $D = S \times T$ dimensional space. During the training phase, we



(a) Evaluation scheme used on the datasets with non overlapping windows.



(b) Evaluation scheme used on the datasets with predefined cuts/folds.

Figure 4.1: Evaluation schemes used to provide the results.

calculate a single representative point for each of the target classes. These points are the element-wise means of all training samples for given class. Classification for a given sample is then performed by calculating the L2 distance to each of the representative points, and the class with the minimal L2 distance to its representative point is chosen as the classification result.

For each of the seven datasets, the algorithm was run 100 times with different random seeds used for splitting into the train (90%) and test (10%) sets. In Figure 4.2 we present a standard box plot, showing the distributions of the metrics.

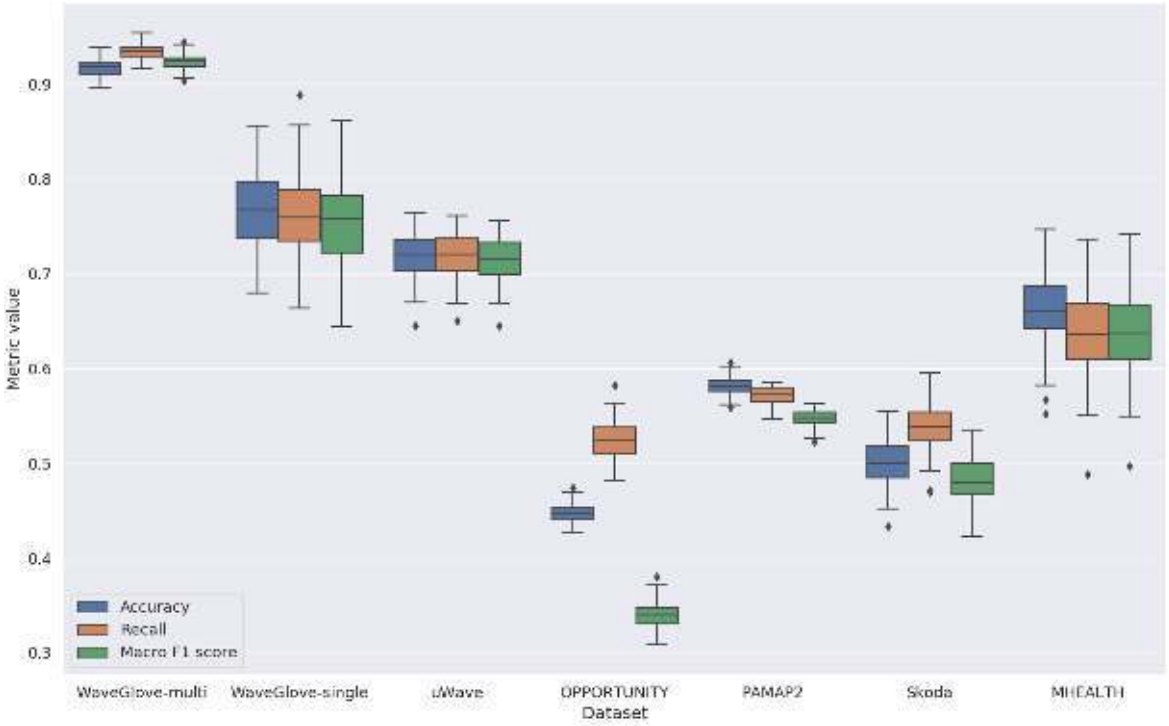


Figure 4.2: Results obtained using 100 runs of the average representative classifier. For each category a box is drawn from the first quartile ($Q1$, 25th percentile) to the third quartile (Q_3, 75th percentile). The line in the box denotes the median value. Afterwards the Interquartile Range - $IQR = Q_3 - Q_1$ is calculated. The whiskers above and below the box are drawn up to the farthest observed value from the data that falls within distance of $1.5IQR$ from the boundaries of the box. The rest of the values are marked as outliers and plotted as separate points [101].

The WaveGlove-multi dataset shows exactly what we wanted to achieve during the design of its gesture vocabulary - the possible space of multi-finger movements is much larger that that of single point movements, and we were successful at unfolding the vocabulary along the space. Such high precision on this naive approach highlights the importance of an ablation study.

The performances of WaveGlove-single and uWave are very comparable due to

the fact that the same vocabulary of gestures was used. This further supports the quality of our dataset as it is comparable to that from previous work. Higher variance in average representative classification of WaveGlove-single is yet to be investigated. On the OPPORTUNITY dataset, which is the most imbalanced of these datasets (75% of samples are from the null class), we can see a major difference between F1 score and the other metrics caused by this imbalance.

The second, more sophisticated non-DL method we examined is a Bagging ensemble of 100 decision tree classifier. During the feature extraction phase, two types of features are used: mean value for each input channel and correlation for every pair of channels. Bagging classifier is implemented using the scikit-learn library[99] and for each decision tree in the ensemble 50% of total samples and 50% of all channels are drawn randomly with replacement for learning. The same technique was employed by [102] and replicated by [8].

Datasets were split for evaluation the same way as with the previous method and for each of the datasets, the algorithm was run on 50 different random seeds used for splitting into the train and test sets. Results are shown in Figure 4.3 and averages of the metrics of both baseline methods are summarized in Table 4.1.

The results obtained using the Bagging Decision tree method suggest that there are fundamental differences in the difficulty of classification of the individual datasets. The median performance on the uWave dataset even dropped compared to the Average representative method, which is very likely due to the longest median length of samples (315). On the WaveGlove-multi, Skoda and MHEALTH datasets, practically all predictions were correct. This likely suggests that the WaveGlove-multi dataset presents a rather simple classification task. On the other hand, results from the following sections suggest that when LOSO or LOTO validation is used on the MHEALTH dataset, the precision is much lower. We can therefore conclude that random split validation hides the user-dependent or trial-dependent differences and can lead to optimistic results when compared to LOSO or LOTO.

4.2 Deep learning methods

Surveys of the state of the art mentioned in Chapter 1 suggest that using neural networks is the most popular deep learning approach to HAR and HGR. For all of these methods, we normalize the input data in a way such that each input channel acts as a random variable with zero mean and unit variance over the whole input. This is considered to be a standard normalization technique for neural network applications.

In the following subsection we describe a simple linear neural network and a network consisting of mostly convolutions. We continue with two models based on using LSTM

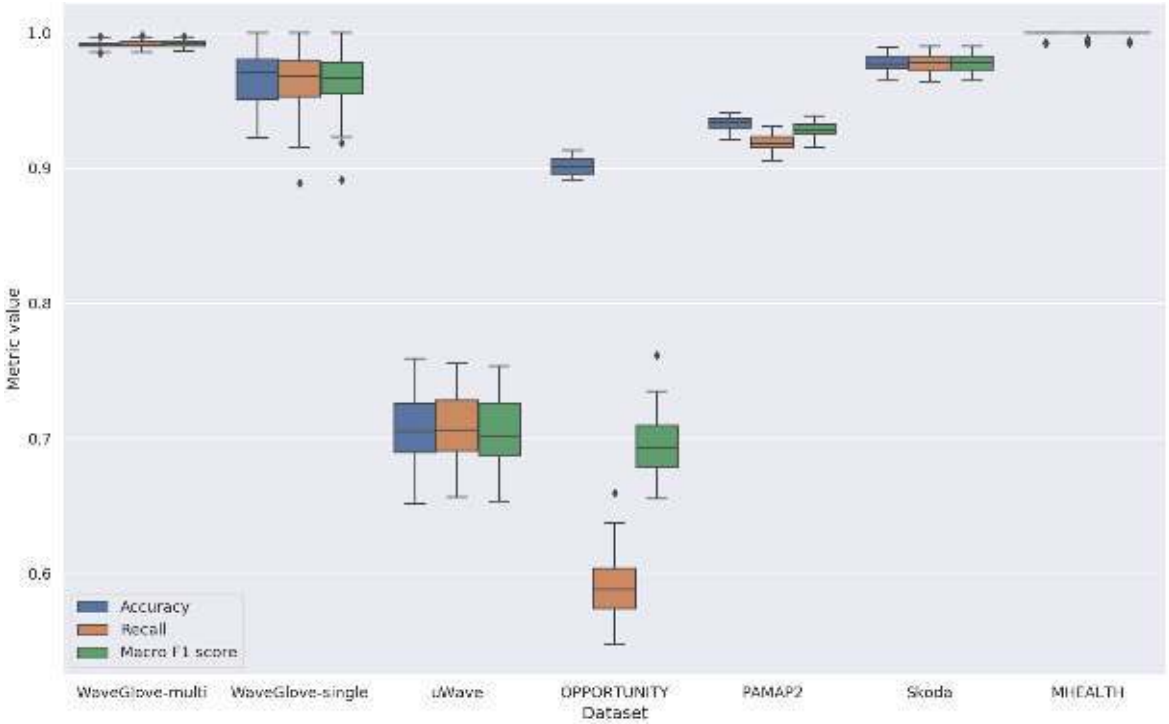


Figure 4.3: Results obtained using 50 runs of the Bagging Decision tree classifier.

Dataset	Average representative			Bagging Decision tree		
	Accuracy	Recall	F1 score	Accuracy	Recall	F1 score
WaveGlove-multi	0.917	0.934	0.923	0.991	0.992	0.992
WaveGlove-single	0.763	0.761	0.753	0.966	0.965	0.964
uWave	0.719	0.720	0.716	0.707	0.708	0.705
OPPORTUNITY	0.447	0.524	0.340	0.902	0.589	0.693
PAMAP2	0.581	0.572	0.547	0.933	0.919	0.929
Skoda	0.501	0.538	0.481	0.978	0.978	0.978
MHEALTH (FNOW)	0.662	0.637	0.637	0.999	0.999	0.999

Table 4.1: Average accuracies, recalls and F1 scores obtained from the baseline methods.

units and two more models from [53] and [95]. Finally, we present a novel Transformer-based, self-attention network architecture. Images showing the network architecture were generated using the Netron[103] application.

4.2.1 Baseline deep neural networks

As the simplest neural network architecture, we use a multi layer perceptron (MLP) - the network consists of three linear (fully connected) layers. The input of size $T \times S$ is flattened before the first layer and the three linear layers use ReLU, tanh and softmax activation functions respectively.

Hyperparameter search has been performed on datasets with non-overlapping windows, using $k = 4$ folds. For the second and third linear the sizes of $64, 128, 256$ and 512 neurons per layer and learning rates of $0.001, 0.0001$ and 0.00001 were searched. Of the resulting 64 combinations, 256 neurons on the second layer, 128 neurons on the third layer and learning rate 0.001 was found to be the best. The final architecture is shown in Figure 4.4a.

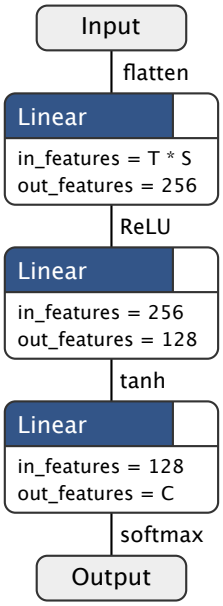
A great way to make a network aware of spatial (or temporal) dependencies is using convolutional layers. Therefore for the second baseline we therefore chose an architecture similar to the of [104]. We changed the architecture to include zero padding before the convolutional layers, so that outputs after convolutions have the same dimensions and inputs before padding. Otherwise, due to maxpooling layers after convolutions this method could not be used on datasets with short temporal sample dimension.

In our architecture, zero padding, two dimensional convolution and maxpooling layers are used three times with a final linear layer to map the outputs to target classes. ReLU activation is used after each of the convolutional layers. First of the convolutional layers has kernel size $(12, 2)$, allowing data from sensors to be spread to neighbouring sensors. The second and third convolutional layers have kernel sizes of $(13, 1)$ and $(12, 1)$ respectively and all of the maxpool layers have kernel size of $(2, 1)$. This way, after the first layer and until the final linear layer, information from each of the sensors is processed separately.

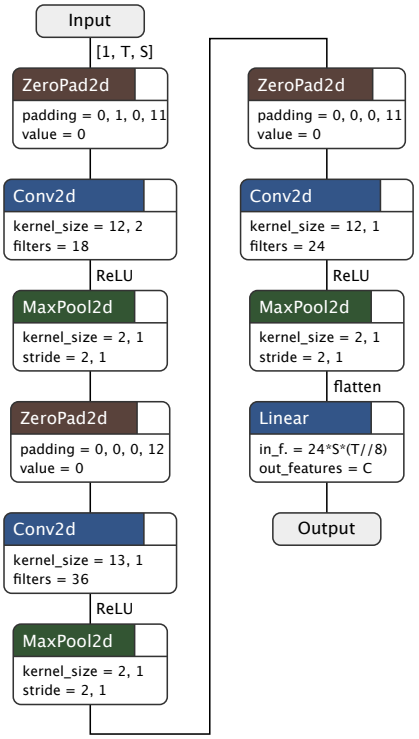
Hyperparameter search with $k = 4$ has again been performed. Filter counts of $9, 18, 36$ for the first two layers, $12, 24$ for the third layer and learning rates $0.001, 0.0001$ were explored. The final architecture (including the best performing hyperparameters) is shown in Figure 4.4b.

4.2.2 LSTM networks

Recurrent neural networks are a popular network architecture that allows to capture temporal dependencies of the input. Similar to previous work, we use the LSTM cells as the basic building block of our RNNs. Our experiments have shown that even



(a) MLP with three linear layers.



(b) A convolutional neural network with three zero-padded convolutional layers and a final linear layer.

Figure 4.4: Baseline DNN models.

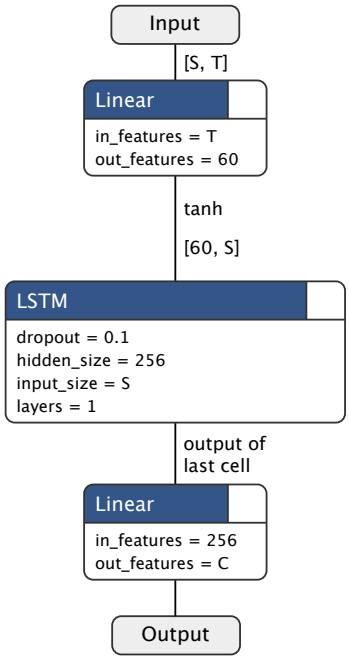
LSTM-based networks are hard to train on sequences with high temporal dimension. In particular, if the temporal dimension of the sample is higher than 60 (which is the case for most of the datasets), we were unable to find a hyperparameter combination which leads to decreasing loss. Moreover, with sequences longer than 60 our networks were not able to overfit a small portion of the training set, leading us to believe that the model needs to be further tweaked.

The first way to allow the LSTM cells to work with shorter sequences, is to learn a linear time embedding (Figure 4.5a). Therefore we add a linear layer before the LSTM layer, which has T input features and 60 output features. Before applying the linear layer, the sample is transposed, so that the same set of weights is used for each of the S input channels. This linear layer can be considered as a learned projection onto temporal length of 60. Hyperbolic tangent activation function is used before the next layer. A single LSTM layer is used, with hidden state size of 256 and 0.1 dropout. Finally, a second linear layer maps the output of the last LSTM cell onto target class labels.

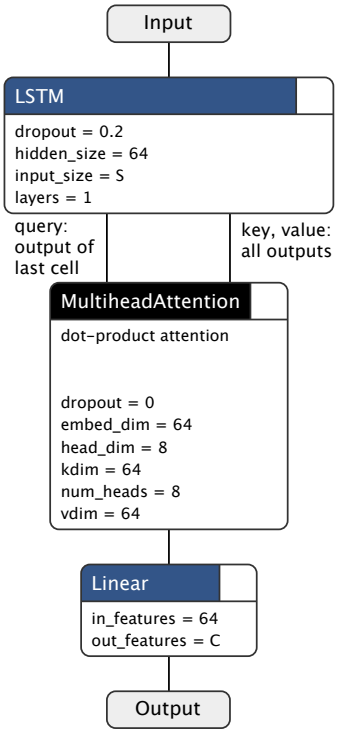
The second LSTM model we show uses self-attention (a technique which gained a lot of attention lately). This allows the model to focus on some of the T recurrent outputs (Figure 4.5b). The LSTM layer has a hidden state size of 64 followed by a

0.2 dropout. The attention mechanism it uses is a simple dot-product attention with 8 heads. Output of the last LSTM cell is used as the query, and outputs of all cells are used as key and value. Finally, a linear layer maps the attention output onto target class labels.

For both models, hyperparameter searches were performed with hidden state sizes $64, 128, 256$ and 512, dropouts $0.1, 0.2$ and 0.3 and time embedding lengths of $30, 50, 60$ and 80 for the first model. The aforementioned combination described above was found to be the best performing one. Learning rate was set to 0.001.



(a) LSTM network with time embedding.



(b) LSTM network with self attention.

Figure 4.5: Models using LSTM layers.

4.2.3 DeepConvLSTM networks

A combination of convolutional and LSTM layers was proposed in [53], naming the architecture DeepConvLSTM. We did reproduce two DeepConvLSTM architectures from previous work in order to compare them with other models.

The first model is reproduced from the description and code published in [53]. It uses four convolutional layers, with 64 filters each and $(5, 1)$ kernel sizes, without padding. Each of the convolutional layers is followed by ReLU activation. The output of the fourth convolutional layer is fed to the first of two LSTM layers with hidden state size of 128. Learning rate was set to 0.001. Finally, the output of the last LSTM cell from the second layer is mapped to target classes using a linear layer. The model is

shown in Figure 4.6a. In the original work, the samples were generated using a short, $500ms$ sliding window, which caused the network to not run into the problems with too long sequences as mentioned above. Therefore the performance on datasets in our work may be lower than that reported by the authors.

In order to improve the overall model performance (most notably on longer sequences), [95] proposed a model using self-attention. We reproduced the model from the description and the published code. The model uses a single convolutional layer with kernel size $(1, S)$ with three filters, effectively mapping the S input channels into 3. This input is fed into a single LSTM layer with hidden size of 32 without dropout. Aftewards, they proposed a self-attention mechanism using two learned weight matrices W_a and W_b. The output of the attention layer is calculated as (with in as the layer input:

$$out = softmax(W_b \cdot tanh(W_a \cdot in^T)) \cdot in$$

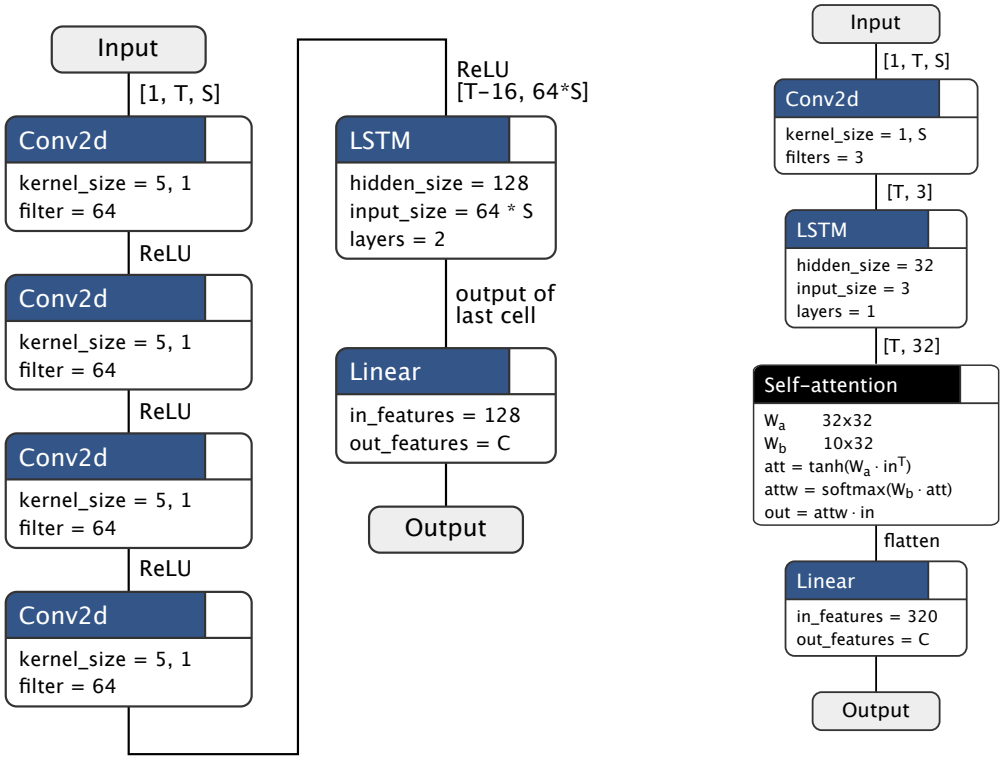
As in other methods a final linear layer maps the features into target class labels. The model is shown in Figure 4.6b, the learning rate was set to 0.0001 as in the original work.

This model was originally evaluated using LOSO and LOTO validation on the datasets provided by [8]. During the inspection of the original code published by [95], we discovered that the evaluation method did use the folds provided with the datasets, but no training and validation split was used. Unfortunately, the authors used the test set for early stopping of the training, effectively making the validation and test sets equal. This suggests extensively skewed results reported in the original work. Our reproduction shows that this model does not outperform most of other models, contradicting the reported results. We note that we did not perform a hyperparameter search on this model, which could improve the situation slightly.

4.2.4 Transformer-based self-attention network

Finally, we present a novel architecture using solely the self-attention mechanism, without recurrent layers. The model is based on the encoder layers of the Transformer introduced in [64].

Firstly a linear layer with S input features and 32 output features learns sensor embedding (same weight matrix is used throughout the temporal window). After this layer an activation $f(x) = x + tanh(x)$ is used. Since our model does not use any convolution or recurrent connections, in order for the model to make use of the order of the sequence, we add positional encoding. We use the same positional encoding as described in [64], adding sine and cosine values of different frequencies. The positional encoding contains an additional dropout of 0.2.



(a) DeepConvLSTM from [53].

(b) DeepConvLSTM with self-attention from [95]

Figure 4.6: Two reproduced DeepConvLSTM architectures.

Four encoder layers follow, which are the same as in the Transformer architecture - a multi-head attention and linear layer both followed by batch normalization and with residual connections. For the linear layers, the size 128 is chosen and 8 heads are used in the self-attention. Again, a 0.2 dropout is introduced.

After the encoder layers a global temporal attention layer is introduced, similar to that in [65]. The layer uses the last output in the temporal dimension as query and all outputs of the previous layer as key and value. A dot product attention without dropout and with 8 heads is used. Finally, a linear layer maps the outputs onto target classes.

Hyperparameter search was performed, we used 16 and 32 for the sensor embedding dimension, 4 and 8 attention head combinations in the encoder and temporal attention layers, 4 and 6 encoder layers and encoder linear layer sizes of 128 and 256. Learning rates of 0.001 and 0.0001 were examined, with the former yielding better results and therefore being used in the final evaluation. The values described above and in Figure 4.7 were found to yield the best results.

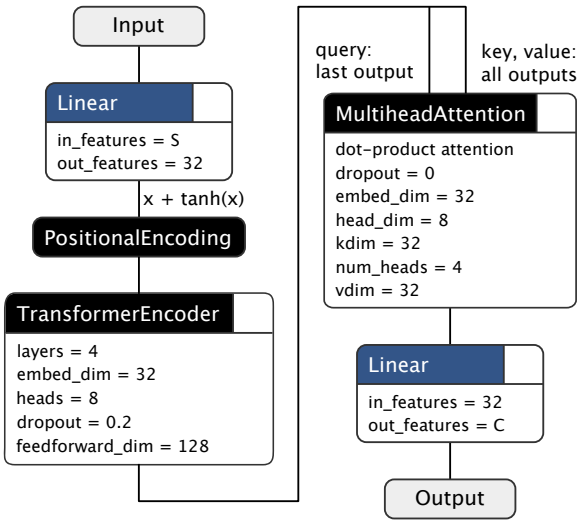


Figure 4.7: The novel, Transformer-based model using no convolutional or recurrent layers.

4.3 Results summary

In this section we summarize the results of the proposed models. First, we compare their performance on the datasets with non-overlapping samples and then we use compare their accuracy to that reported by previous work on the datasets with LOSO and LOTO predefined folds.

We report accuracy, recall and F1 score for the DL models described in the previous sections, on the non-overlapping sample datasets in the Figure 4.8. DeepConvLSTM with self-attention was left out, as it has shown consistently worse performance than the original DeepConvLSTM model in our experiments.

Three HGR datasets (WaveGlove datasets and uWave dataset) provide consistently better performance than other HAR datasets. This supports the claim that classifying movements performed of the hand with one or more IMUs, presents a lesser challenge than classifying whole body movements.

OPPORTUNITY and PAMAP2 are the two most imbalanced datasets. All of the proposed methods achieved the worst performance on the OPPORTUNITY dataset. The high difficulty of classification in this dataset may be caused by very little inter-class differences between some of the activities, e.g. opening drawer 1 and opening drawer 2. In PAMAP2, the activities are not only imbalanced between each other, but also between the subjects. Some of the activities were performed only by one of the subjects as shown in Table B.1, which can also contribute to the difficulty.

The poor to fatal results of the DeepConvLSTM architecture are likely caused by improper choice of various hyperparameters. We did use the model with hyperparameters reported in [53] without further tuning. The authors did present benchmarks on

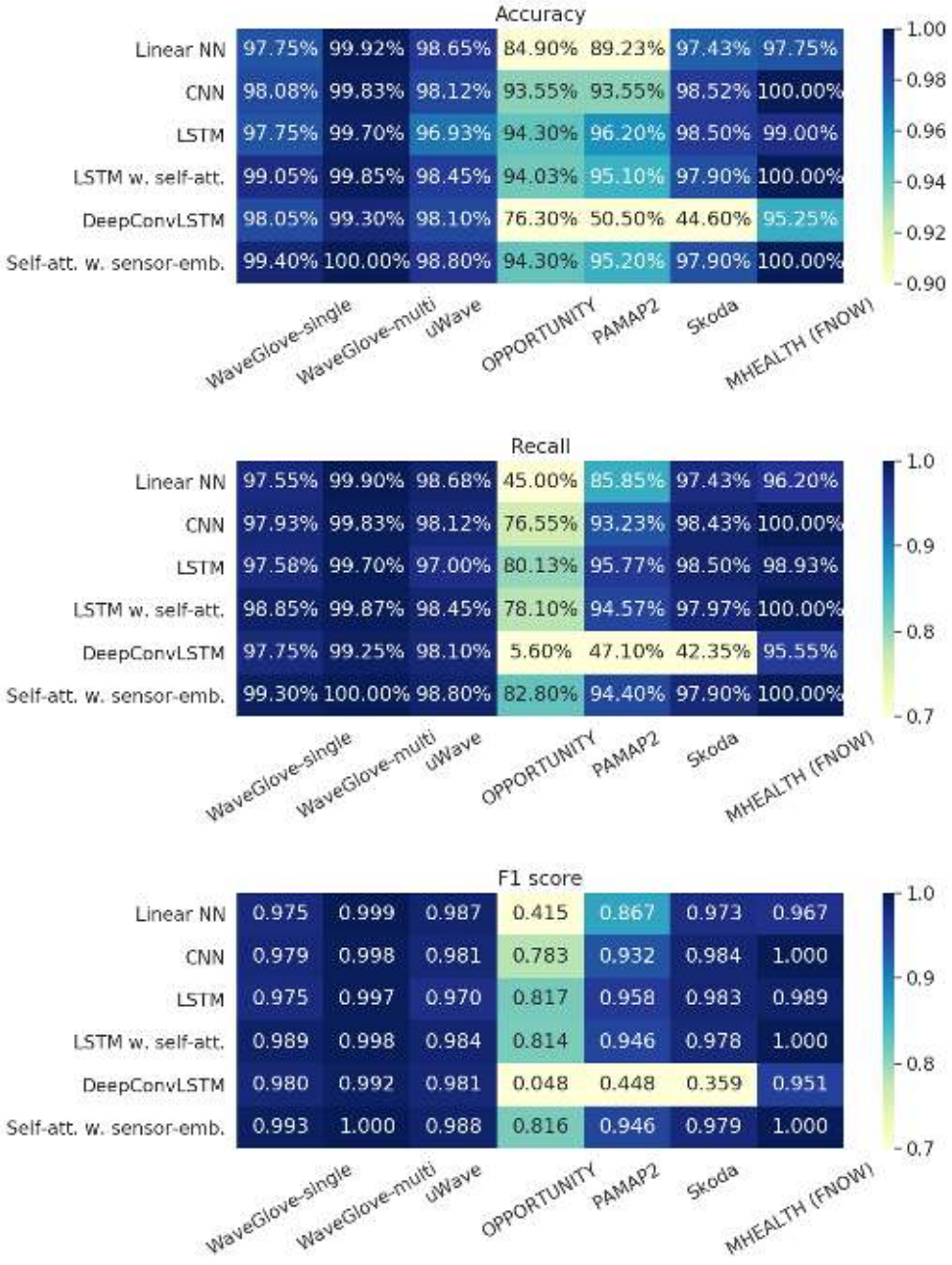


Figure 4.8: Accuracy, recall and F1 scores of models evaluated on the datasets with non-overlapping samples. Reported metrics represent the average of 4 runs (differing in the random test split), with the exception of the DeepConvLSTM and Self-attention with sensor-embedding networks, for which only 2 runs were performed.

the OPPORTUNITY and SKODA datasets, however the method they used for sample generation was different.

Near perfect classification was achieved by the CNN, LSTM with self-attention and the Transformer-based using self-attention models on the MHEALTH dataset. Based on the benchmarks presented in the next section, we can conclude that splitting the MHEALTH dataset randomly into the training and test sets can lead to overly optimistic results. In the next section, the dataset is split using LOTO and LOSO methods, changing the results considerably.

Our newly proposed model achieved the best accuracy on five out of the seven datasets. However, it's overall performance is very similar to the two LSTM-based models.

In [8], the authors provide a set of six datasets split into training and testing sets in four different ways. In Tables 4.2 and 4.3 we show a comprehensive comparison of accuracies of several classification methods on the LOTO and LOSO splits. We include methods:

- Re-implemented in [8].
- Proposed in [100].
- Re-implemented by us and originally proposed in [53] and [95].
- Proposed in our work.

The results suggest that LOSO validation presents a higher challenge than LOTO validation. This is likely caused by higher differences between the subjects. Moreover, by splitting the MHEALTH dataset using the LOTO or LOSO protocols, the methods we propose show substantially worse results. Therefore, for further evaluation and research, using LOTO or LOSO rather than the windowing method we proposed seems more viable, as they simulate a more realistic scenario.

The MHEALTH dataset contains the most sensor channels (23) from the datasets shown in the tables. When using the Bagging Decision Tree classifier, the number of features used grows with the number of channels quadratically, which may be the reason for its outstanding performance on the dataset.

Using the attention mechanism in combination with LSTM layers as proposed does consistently increase classification performance when compared with the proposed time embedding method. We were not able to train LSTMs using the longer time sequences without any embedding or attention mechanism.

Finally, the proposed Transformer-based self-attention network architecture shows promising results, comparable to those of the LSTM network using the self attention mechanism. We believe that further investigation and hyperparameter tuning may further improve its performance.

The method proposed in [100] using ensembles of CNN classifiers demonstrate exceptional performance on some of the datasets as well. None of the compared methods dominates the others on all of the datasets.

Method	LOTO (accuracy (%))						Avg.
	MHEALTH	**USC-HAD**	**UTD-MHAD1**	**UTD-MHAD2**	**WHARF**	**WISDM**	
Kwapisz et al. [46]	89.75	76.52	15.99	69.61	44.51	79.08	62.57
Catal et al. [105]	91.84	87.77	47.80	81.37	64.84	80.52	75.69
Kim et al. [102]	91.51	85.70	50.98	75.27	61.12	56.26	70.14
Chen and Xue [104]	89.95	84.66	-	-	72.55	86.55	83.42
Jiang and Yin [106]	52.78	80.73	-	-	70.79	83.82	72.03
Ha et al. [107]	85.31	-	-	-	-	-	85.31
Ha and Choi [108]	82.75	-	-	-	-	-	82.75
Nguyen et al. [45]	**94.72**	86.90	-	-	-	-	90.81
Sena and Schwartz [100]	93.09	88.49	62.03	81.63	75.50	89.01	81.62
DeepConvLSTM [53]	81.01	83.97	67.29	86.50	67.98	**91.23**	79.66
DeepConvLSTM with attention[95]	74.08	78.64	48.31	74.15	65.31	86.57	71.17
Proposed models							
Bagging Decision tree	**93.41**	89.22	67.51	81.90	66.26	61.61	76.65
CNN	87.35	**89.87**	62.74	83.47	78.03	**90.59**	82.00
LSTM with time embedding	81.69	82.96	63.61	72.00	67.76	79.34	74.56
LSTM with attention	86.44	89.79	69.72	84.13	**81.47**	90.07	83.60
Self-attention with sensor embedding	90.35	89.83	**76.32**	**88.42**	78.63	84.53	**84.68**

Table 4.2: Evaluation of the proposed and previous methods on LOTO predefined folds. The "-" symbol denotes that the given method could not be applied on the dataset.

Method	LOSO (accuracy (%))						Avg.
	MHEALTH	**USC-HAD**	**UTD-MHAD1**	**UTD-MHAD2**	**WHARF**	**WISDM**	
Kwapisz et al. [46]	90.41	70.15	13.04	66.67	42.19	75.31	59.62
Catal et al. [105]	94.66	75.89	32.45	74.67	46.84	74.96	66.57
Kim et al. [102]	93.90	64.20	38.05	64.60	51.48	50.22	60.40
Chen and Xue [104]	88.67	75.58	-	-	61.94	83.89	77.52
Jiang and Yin [106]	51.46	74.88	-	-	65.35	79.97	67.91
Ha et al. [107]	88.34	-	-	-	-	-	88.34
Ha and Choi [108]	84.23	-	-	-	-	-	84.23
Sena and Schwartz [100]	**96.27**	82.66	46.75	**79.38**	**69.79**	86.22	**76.85**
DeepConvLSTM [53]	83.58	70.96	48.42	75.65	61.23	84.43	70.71
DeepConvLSTM with attention[95]	74.18	70.12	42.57	69.20	63.97	81.06	66.85
Proposed models							
Bagging Decision tree	**94.53**	72.66	49.45	73.28	60.09	56.19	67.70
CNN	91.05	77.15	47.62	76.45	66.67	**86.58**	74.25
LSTM with time embedding	85.61	77.31	45.33	68.16	62.73	76.29	69.24
LSTM with attention	89.23	**80.67**	52.05	78.55	**68.53**	83.26	**75.38**
Self-attention with sensor embedding	91.04	73.44	**55.85**	**79.19**	63.05	74.90	72.91

Table 4.3: Evaluation of the proposed and previous methods on LOSO predefined folds. The "-" symbol denotes that the given method could not be applied on the dataset.

4.4 Ablation study on the WaveGlove datasets

In the previous section, nearly all classification methods were shown to have near-perfect accuracy and other metrics on the WaveGlove datasets. Due to these results and the multi-sensor structure of the WaveGlove datasets, an ablation study is warranted in order to asses the impact of various factors on the final performance. In this ablation study we examine the impact of two factors on the classification accuracy. Firstly, we review the accuracy changes resulting from the usage of a subset of the sensors. Secondly, we examine the impact of the size of the training set.

There are a total of 31 non-empty subsets of the five sensors (including the full setup). We group them by the amount of sensors they contain into five groups. There are 5 subsets containing a single sensor and $10, 10, 5$ and 1 subset containing $2, 3, 4$ and 5 sensors respectively. Various training set sizes are used: $1\%, 5\%, 20\%, 50\%$ and 85%, with the rest of the samples selected for testing.

Two methods are used during the ablation study: the Bagging Decision Tree model and the proposed Transformer-based network. Figures 4.9 and 4.10 show the results on the WaveGlove-single and WaveGlove-multi datasets respectively.

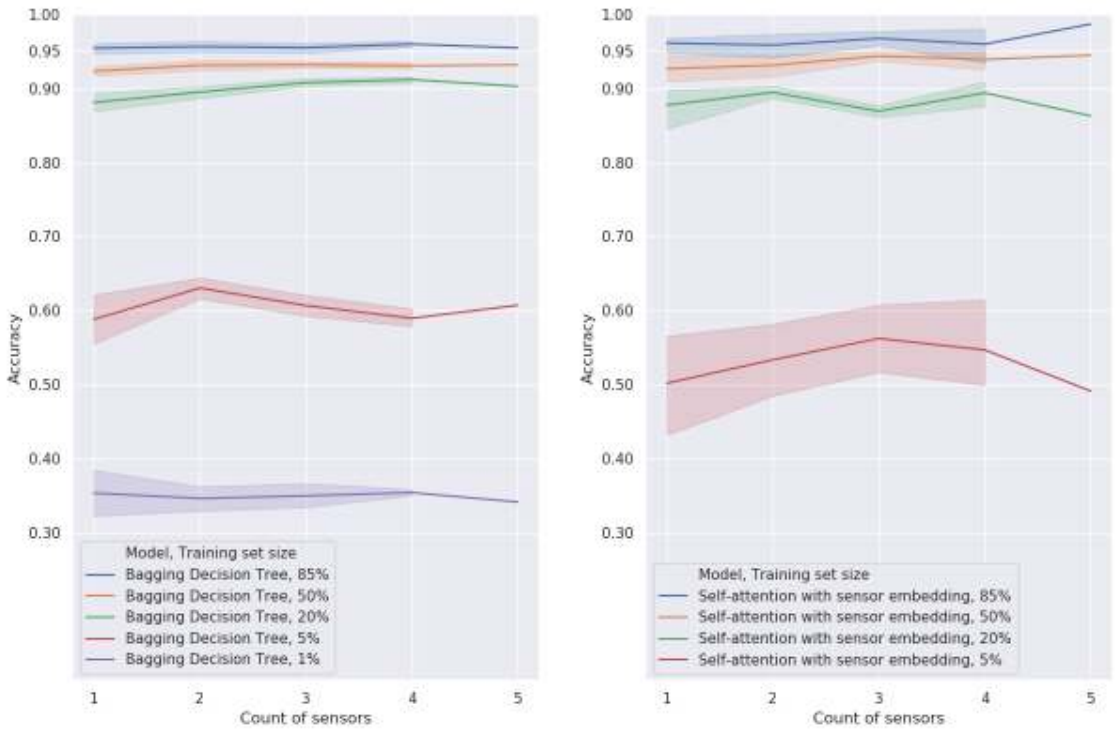


Figure 4.9: Attribution of the count of sensors and training set size to the accuracy on the WaveGlove-single dataset. Training set of size 1% was not sufficient to train the self-attention based model.

Results of the experiment on the WaveGlove-single dataset suggest that increasing the number of used sensors does not have significant impact on classification ac-

curacy in this dataset. The outcome does not come as a surprise, as all eight gestures in the vocabulary are whole hand movements - all of the fingers perform the same movement. Furthermore, training set sizes of 50% and 20% (500 and 200 samples approximately) achieve comparable accuracy to the standard setup of using 85% of the samples in the training set.

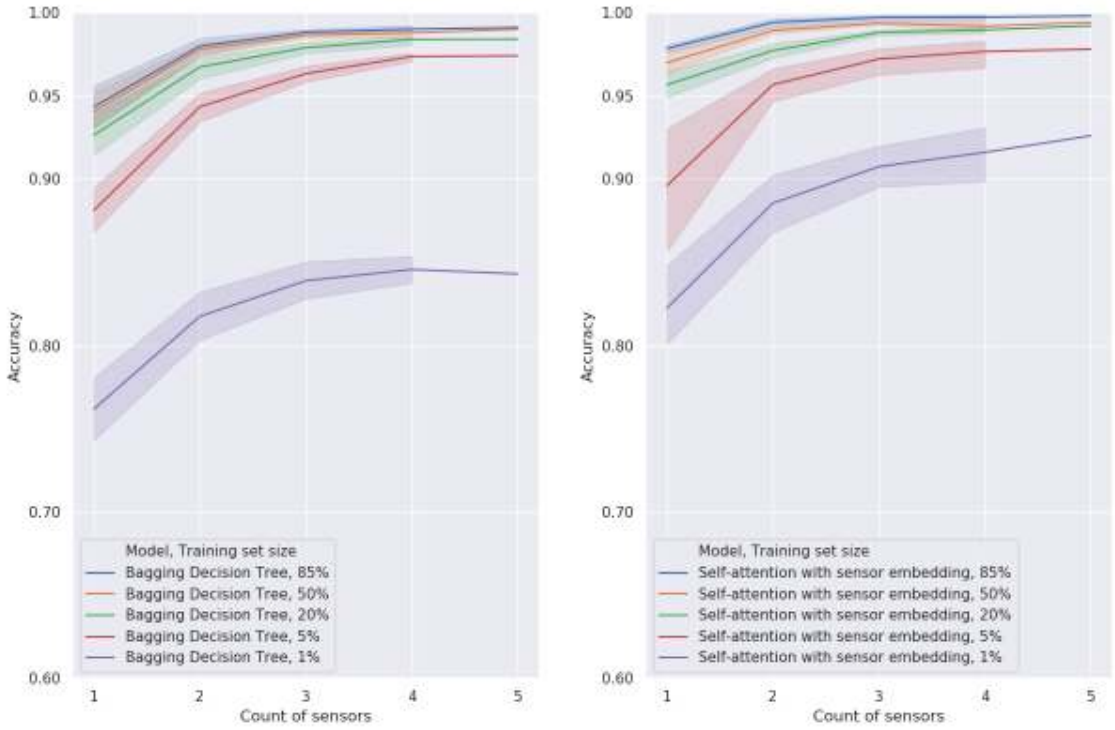


Figure 4.10: Attribution of the count of sensors and training set size to the accuracy on the WaveGlove-multi dataset.

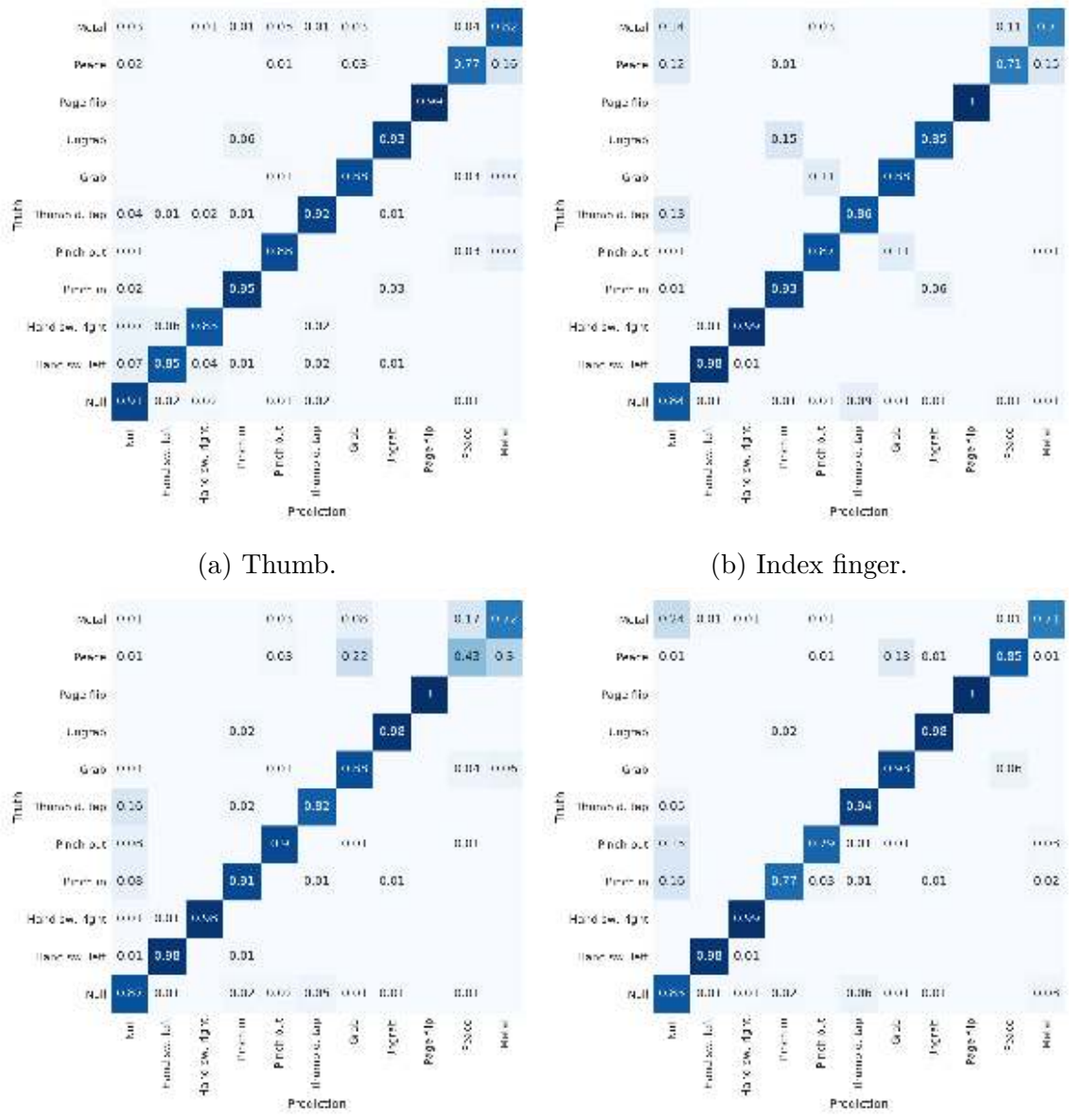
On the other hand, in the case of the WaveGlove-multi dataset, increasing the number of sensors does increase the classification accuracy significantly.

The vocabulary of the dataset was designed to specifically contain movements using only some of the fingers, leading to the expected result. Similar to WaveGlove-single, using a training set of size 20% or even 5% (approximately 2000 and 500 samples respectively), shows very similar results in comparison to the standard setup.

To examine the differences in classification when not using all of the sensors, we compare the four confusion matrices shown in Figure 4.11. The confusion matrices were obtained using the Bagging Decision Tree classifier on the WaveGlove-multi dataset with a training set containing 5% of all samples. In each case, only values from a single sensor were used, corresponding to four fingers: thumb, index finger, ring finger and pinky.

Results support the intuitive claim, that gestures where a finger performs a similar movement are harder to classify using only the sensor on that finger. When using the

sensor on the ring finger, the *Peace* gesture was classified with only 43% accuracy. This is caused by very similar movements performed by the ring finger during the *Grab*, *Peace* and *Metal* gestures. Similarly, the sensor on the thumb turns out to be less effective in distinguishing between the *Hand Swipe Left* and the *Hand Swipe Right* gestures and the sensor on the index finger in distinguishing between the *Pinch Out* (*Pinch In*) and *Grab* (*Ungrab*) gestures. The pinky does not perform any movement during the *Pinch in* and *Pinch out* gestures, resulting in reduced accuracy when using only the corresponding sensor alone.



(a) Thumb.

(b) Index finger.

(c) Ring finger.

(d) Pinky.

Figure 4.11: Four confusion matrices obtained using the Bagging Decision Tree classifier on the WaveGlove-multi dataset. Only data from a single sensor on four different fingers was used.

Conclusions

In our work, we investigate the challenges of Human Activity Recognition (HAR) and Hand Gesture Recognition (HGR) using several methods evaluated on over 10 datasets.

We built a specialized hardware prototype called WaveGlove in the form of a glove. A single IMU is mounted on each of the fingers of the glove , allowing to record multi-sensor samples. We created custom firmware, and designed an efficient protocol for the communication between a microcontroller connected to the prototype and a recording device (usually a laptop).

We used two hand gesture dataset vocabularies. The first one is identical to that used in [15], with the difference that we use more than one sensor in our setup. The second one contains 10 gestures and is specifically designed to contain gestures with different movements for each of the fingers. Using the prototype we record two datasets: WaveGlove-single and WaveGlove-multi of sizes 1000 and 10000 respectively. We consider the size of the datasets to be comparable to that of the previously published HAR and HGR datasets. The only known major disadvantage of the WaveGlove datasets is that only 2 test subjects participated. Unfortunately, the pandemic situation in the spring of 2020 did not allow the participation of more subjects.

For the performance evaluation of classification methods, we adopted several different publicly available datasets. We preprocessed some of them using a proposed sample generation technique. Furthermore, 6 datasets preprocessed in [8] were used with the same exact train and test set splits, for increased comparability.

We implemented 9 different classification models, including both Classical Machine Learning and Deep Learning methods. The performance of the methods ranges from baseline to state of the art. We propose a novel neural network architecture, based on the Transformer architecture introduced in [64]. This newly proposed method does not use recurrent layers, yet achieves comparable or arguably better performance on some of the datasets. In the LOTO evaluation our method achieved mean accuracy 3.06% higher than that reported in previous work. It's accuracy was also 8.03% higher than that of the Bagging Decision Tree method and 1.08% than that of the LSTM model using attention.

Using our evaluation of the methods on multiple datasets, we can conclude that the sample generation process can fundamentally affect the final classification performance.

In particular, the methods evaluated on the MHEALTH have experienced $5 - 15\%$ difference in accuracy, depending on the used sample generation process.

We reproduce the DeepConvLSTM-based methods proposed in [53] and [95]. With our proposed methods, we also heavily extend the standardization benchmarks established in [8].

Based on our results, we conclude that methods achieve consistently better performance on the HGR datasets in comparison to the considered HAR datasets.

Finally, we perform an ablation study on the WaveGlove datasets. Our study shows, that increasing the number of sensors in the WaveGlove-single dataset does not increase the performance significantly. This is in line with the expectations, because identical movement is executed by each of the fingers for the gestures in the corresponding vocabulary. On the other hand, we show that for the WaveGlove-multi dataset, increasing the number of sensors does increase the classification accuracy. Major improvements are observed while using up to 3 sensors, with lesser to no improvements obtained from adding the fourth or fifth sensor.

We show that a smaller training set consisting of a few hundred samples is sufficient for achieving near the same performance as with over 8500 samples for the WaveGlove-multi dataset. By comparing the confusion matrices of classification methods using various subsets of the five sensors on the WaveGlove, we confirm that different gestures can be classified with higher performance using different subsets of sensors.

For future work, we recommend closer investigation of the different dataset splitting methods and their effect on final performance. Splitting techniques which model the final practical use case should be preferred. The LSTM model using self-attention as well as the model using self-attention with sensor embedding have shown very promising performance. We suggest further fine tuning and hyperparameter search for the attention-based models, to achieve even higher classification accuracy. Due to the fact that the WaveGlove-single dataset has identical vocabulary to the dataset published in [15], we advise the examination of the possible application of transfer learning. In the case of a practical application of a WaveGlove-like device, we recommend the use of wireless communication and the problem of distinguishing between gesture and non-gesture states to be further investigated.

Bibliography

[1] Matej Andrejašic. Mems accelerometers. In *University of Ljubljana. Faculty for mathematics and physics, Department of physics, Seminar*, 2008.

[2] Chaithanya Kumar Mummadi, Frederic Philips Peter Leo, Keshav Deep Verma, Shivaji Kasireddy, Philipp M. Scholl, Jochen Kempfle, and Kristof Van Laerhoven. Real-time and embedded detection of hand gestures with an imu-based glove. *Informatics*, 5(2):28, 2018.

[3] Casey A. Cole, Bethany Janos, Dien Anshari, James F. Thrasher, Scott M. Strayer, and Homayoun Valafar. Recognition of smoking gesture using smart watch technology. *CoRR*, abs/2003.02735, 2020.

[4] Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, and Parisa Rashidi. Human activity recognition using inertial, physiological and environmental sensors: a comprehensive survey. *CoRR*, abs/2004.08821, 2020.

[5] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Syst. Appl.*, 105:233–261, 2018.

[6] Marcin Straczkiewicz and Jukka-Pekka Onnela. A systematic review of human activity recognition using smartphones. *CoRR*, abs/1910.03970, 2019.

[7] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges and opportunities. *CoRR*, abs/2001.07416, 2020.

[8] Artur Jordao, Antonio C. Nazare Jr., Jessica Sena de Souza, and William Robson Schwartz. Human activity recognition based on wearable sensor data: A standardization of the state-of-the-art. *CoRR*, abs/1806.05226, 2018.

[9] Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, and Parisa Rashidi. Human activity recognition using inertial, physiological and environmental sensors: a comprehensive survey. *CoRR*, abs/2004.08821, 2020.

[10] Ian Cleland, Basel Kikhia, Chris D. Nugent, Andrey Boytsov, Josef Hallberg, Kåre Synnes, Sally I. McClean, and Dewar D. Finlay. Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, 13(7):9183–9200, 2013.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning, 2018.

[13] Mridul Khan, Sheikh Iqbal Ahamed, Miftahur Rahman, and Ji-Jiang Yang. Gesthaar: An accelerometer-based gesture recognition method and its application in nui driven pervasive healthcare. In *ESPA*, pages 163–166, 2012.

[14] Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, and Shijian Li. Gesture recognition with a 3-d accelerometer. In *International Conference on Ubiquitous Intelligence and Computing*, pages 25–38. Springer, 2009.

[15] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[16] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14. ACM, 2008.

[17] Diman Zad Tootaghaj, Adrian Sampson, Todd Mytkowicz, and Kathryn S McKinley. High five: Improving gesture recognition by embracing uncertainty. *arXiv preprint arXiv:1710.09441*, 2017.

[18] Steven W Smith et al. *The scientist and engineer's guide to digital signal processing*. California Technical Pub. San Diego, 1997.

[19] Ahmad Akl and Shahrokh Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2270–2273. IEEE, 2010.

[20] Ce Li, Chunyu Xie, Baochang Zhang, Chen Chen, and Jungong Han. Deep fisher discriminant learning for mobile hand gesture recognition. *Pattern Recognition*, 77:276–288, 2018.

[21] Michael Xie and David Pan. Accelerometer gesture recognition, 2014.

[22] David Mace, Wei Gao, and Ayse Coskun. Accelerometer-based hand gesture recognition using feature weighted naïve bayesian classifiers and dynamic time warping. In *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, pages 83–84. ACM, 2013.

[23] David Mace, Wei Gao, and Ayse K Coskun. Improving accuracy and practicality of accelerometer-based hand gesture recognition. *on Interacting with Smart Objects*, page 45, 2013.

[24] Zhenyu He, Lianwen Jin, Lixin Zhen, and Jiancheng Huang. Gesture recognition based on 3d accelerometer for cell phones interaction. In *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 217–220. IEEE, 2008.

[25] Amara Graps. An introduction to wavelets. *IEEE computational science and engineering*, 2(2):50–61, 1995.

[26] Gabriele Costante, Lorenzo Porzi, Oswald Lanz, Paolo Valigi, and Elisa Ricci. Personalizing a smartwatch-based gesture interface with transfer learning. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 2530–2534. IEEE, 2014.

[27] Matthias Rehm, Nikolaus Bee, and Elisabeth André. Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1*, pages 13–22. British Computer Society, 2008.

[28] Ahmed Hassan. Classification of accelerometer data as input for user interfaces, 2007.

[29] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[30] Prajwal Paudyal, Ayan Banerjee, and Sandeep KS Gupta. Sceptre: a pervasive, non-invasive, and programmable gesture recognition technology. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 282–293. ACM, 2016.

[31] RMW Kluge. Online accelerometer gesture recognition using dynamic time warping and k-nearest neighbors clustering with flawed templates. Bachelor thesis, 2017.

[32] Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. Sparsedtw: A novel approach to speed up dynamic time warping. *CoRR*, abs/1201.2969, 2012.

[33] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[34] Johann Faouzi and Hicham Janati. pyts: A python package for time series classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020.

[35] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International journal of pattern recognition and artificial intelligence*, 15(01):9–42, 2001.

[36] Marcus Georgi, Christoph Amma, and Tanja Schultz. Recognizing hand and finger gestures with IMU based motion and EMG based muscle activity sensing. In Harald Loose, Ana L. N. Fred, Hugo Gamboa, and Dirk Elias, editors, *BIOSIGNALS 2015 - Proceedings of the International Conference on Bio-inspired Systems and Signal Processing, Lisbon, Portugal, 12-15 January, 2015*, pages 99–108. SciTePress, 2015.

[37] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[38] Hobeom Han and Sang Won Yoon. Gyroscope-based continuous human hand gesture recognition for multi-modal wearable input device for human machine interaction. *Sensors*, 19(11):2562, 2019.

[39] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

[40] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.

[41] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[42] Antonio A. Aguileta, Ramon F. Brena, Oscar Mayora, Erik Molino-Minero-Re, and Luis A. Trejo. Multi-sensor fusion for activity recognition—a survey. *Sensors*, 19(17):3808, Sep 2019.

[43] Sijie Zhuo, Lucas Sherlock, Gillian Dobbie, Yun Sing Koh, Giovanni Russello, and Danielle M. Lottridge. Real-time smartphone activity classification using inertial sensors - recognition of scrolling, typing, and watching videos while sitting or walking. *Sensors*, 20(3):655, 2020.

[44] Haodong Guo, Ling Chen, Liangying Peng, and Gencai Chen. Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble. In Paul Lukowicz, Antonio Krüger, Andreas Bulling, Youn-Kyung Lim, and Shwetak N. Patel, editors, *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016, Heidelberg, Germany, September 12-16, 2016*, pages 1112–1123. ACM, 2016.

[45] Huu Du Nguyen, Kim Phuc Tran, Xianyi Zeng, Ludovic Koehl, and Guillaume Tartare. Wearable sensor data based human activity recognition using machine learning: A new approach. *CoRR*, abs/1905.03809, 2019.

[46] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explorations*, 12(2):74–82, 2010.

[47] Michael A. Nielsen. Neural networks and deep learning, 2018.

[48] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, page 9–50, Berlin, Heidelberg, 1998. Springer-Verlag.

[49] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[51] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognit.*, 15(6):455–469, 1982.

[52] Antonio Bevilacqua, Kyle MacDonald, Aamina Rangarej, Venessa Widjaya, Brian Caulfield, and M. Tahar Kechadi. Human activity recognition with convolutional neural networks. In Ulf Brefeld, Edward Curry, Elizabeth Daly, Brian MacNamee, Alice Marascu, Fabio Pinelli, Michele Berlingerio, and Neil Hurley, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part III*, volume 11053 of *Lecture Notes in Computer Science*, pages 541–552. Springer, 2018.

[53] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[54] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 851–860, New York, NY, USA, 2016. Association for Computing Machinery.

[55] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.

[56] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6645–6649. IEEE, 2013.

[57] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[58] Alessandro Carfì, Carola Motolese, Barbara Bruno, and Fulvio Mastrogiovanni. Online human gesture recognition using recurrent neural networks and wearable sensors. In *27th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2018, Nanjing, China, August 27-31, 2018*, pages 188–195. IEEE, 2018.

[59] Yu Guan and Thomas Plötz. Ensembles of deep LSTM learners for activity recognition using wearables. *IMWUT*, 1(2):11:1–11:28, 2017.

[60] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.

[61] Felix A. Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, July 24-27, 2000, Volume 3*, pages 189–194. IEEE Computer Society, 2000.

[62] Philipp Koch, Mark Dreier, Marco Maaß, Martina Böhme, Huy Phan, and Alfred Mertins. A recurrent neural network for hand gesture recognition based on accelerometer data. *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2019, 07 2019.

[63] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[65] Saif Mahmud, M. Tanjid Hasan Tonmoy, Kishor Kumar Bhaumik, A. K. M. Mahbubur Rahman, M. Ashraful Amin, Mohammad Shoyaib, Muhammad Asif Hossain Khan, and Amin Ahsan Ali. Human activity recognition from wearable sensor data using self-attention. *CoRR*, abs/2003.09018, 2020.

[66] Paul Lukowicz, Gerald Pirkl, David Bannach, Florian Wagner, Alberto Calatroni, Kilian Förster, Thomas Holleczek, Mirco Rossi, Daniel Roggen, Gerhard Tröster, Jakob Doppler, Clemens Holzmann, Andreas Riener, Alois Ferscha, and Ricardo Chavarriaga. Recording a complex, multi modal activity data set for context recognition. In Michael Beigl and Francisco Javier Cazorla-Almeida, editors, *ARCS '10 - 23th International Conference on Architecture of Computing Systens 2010, Workshop Proceedings, February 22-23, 2010, Hannover, Germany*, pages 161–166. VDE Verlag, 2010.

[67] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura, and José del R. Millán. Collecting complex activity datasets in highly rich networked sensor environments. In *Seventh International Conference on Networked Sensing Systems, INSS 2010, Kassel, Germany, June 15-18, 2010*, pages 233–240. IEEE, 2010.

[68] Leire Francés, Paz Morer, Maria Isabel Rodriguez, and Aitor Cazón. Design and development of a low-cost wearable glove to track forces exerted by workers in car assembly lines. *Sensors*, 19(2):296, 2019.

[69] Sonu Agarwal, Arindam Mondal, and Gurdeepak Joshi. Gestglove: a wearable device with gesture based touchless interaction. In Pranav Mistry, Pattie Maes, Jean-Marc Seigneur, Suranga Nanayakkara, and Joe Paradiso, editors, *Proceedings of the 8th Augmented Human International Conference, AH 2017, Mountain View, CA, USA, March 16-18, 2017*, page 3. ACM, 2017.

[70] Henry Friday Nweke, Ying Wah Teh, Ghulam Mujtaba, Uzoma Rita Alo, and Mohammed Ali Al-garadi. Multi-sensor fusion based on multiple classifier systems for human activity identification. *HCIS*, 9:34, 2019.

[71] Marcin Straczkiewicz, Nancy W. Glynn, and Jaroslaw Harezlak. On placement, location and orientation of wrist-worn tri-axial accelerometers during free-living measurements. *Sensors*, 19(9):2095, 2019.

[72] Ruize Xu, Shengli Zhou, and Wen J Li. Mems accelerometer based nonspecific-user hand gesture recognition. *IEEE sensors journal*, 12(5):1166–1173, 2012.

[73] Vasileios Sideridis, Andrew Zacharakis, George Tzagkarakis, and Maria Papadopouli. Gesturekeeper: Gesture recognition for controlling devices in iot environments. *CoRR*, abs/1903.06643, 2019.

[74] Mimu gloves - the world's most advanced wearable musical instrument. `https://mimugloves.com`. Accessed: 2020-05-07.

[75] Senseglove - make the digital feel real. `https://www.senseglove.com`. Accessed: 2020-05-07.

[76] Neomano - a robotic glove that enables people with hand paralysis to complete daily activities. `https://neomano.neofect.com`. Accessed: 2020-05-07.

[77] Espressif Systems. *ESP8266EX - Espressif's highly integrated Wi-Fi SoC solution.*, 2020. Version 6.4.

[78] InvenSense Inc. *MPU-6000 and MPU-6050 Product Specification*, 2013. Revision 3.4.

[79] NanJing Top Power ASIC Corp. *TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8.*

[80] Texas Instruments. *TCA9548A - Low-Voltage 8-Channel I2C Switch with Reset.*, 2019. SCPS207G – MAY 2012 – REVISED NOVEMBER 2019.

[81] KylinChip Electronic (Shanghai) Co.,Ltd. *XL6009 - 400KHz 60V 4A Switching Current Boost / Buck-Boost / Inverting DC/DC Converter.*

[82] SHENZHEN PKCELL BATTERY CO., LTD. *Techical Specification Li-Polymer 503035 500mAh 3.7V with PCM*, 2014. QA.S.0228.

[83] Linear Technology Corporation. *LTC3426 - 1.2MHz Step-Up DC/DC Converter in SOT-23.*, 2004. LT 0617 REV B.

[84] Platformio - a new generation ecosystem for embedded development. `https://platformio.org`. Accessed: 2020-05-07.

[85] PyQT. Pyqt reference guide, 2012.

[86] Emiro De la Hoz, Paola Ariza, J. Medina, and Macarena Espinilla. Sensor-based datasets for human activity recognition – a systematic review of literature. *IEEE Access*, PP:1–1, 10 2018.

[87] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In Roberto Verdone, editor, *Wireless Sensor Networks, 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008, Proceedings*, volume 4913 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2008.

[88] Thomas Stiefmeier, Daniel Roggen, and Gerhard Tröster. Fusion of string-matched templates for continuous activity recognition. In *11th IEEE International Symposium on Wearable Computers (ISWC 2007), October 11-13, 2007, Boston, MA, USA*, pages 41–44. IEEE Computer Society, 2007.

[89] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *16th International Symposium on Wearable Computers,*

ISWC 2012, Newcastle, United Kingdom, June 18-22, 2012, pages 108–109. IEEE Computer Society, 2012.

[90] The hdf5® library & file format. `https://www.hdfgroup.org/solutions/hdf5/`, 2007. Accessed: 2020-05-09.

[91] Andrew Collette and contributors. Hdf5 for python. `http://www.h5py.org`, 2008. Accessed: 2020-05-09.

[92] Analog Devices, Inc. *3-Axis,* $\pm 2g/\pm 4g/\pm 8g/\pm 16g$ *Digital Accelerometer*, 2015. Rev. E.

[93] Oresti Baños, Rafael García, Juan Antonio Holgado Terriza, Miguel Damas, Héctor Pomares, Ignacio Rojas Ruiz, Alejandro Saez, and Claudia Villalonga. mhealthdroid: A novel framework for agile development of mobile health applications. In Leandro Pecchia, Liming Luke Chen, Chris D. Nugent, and José Bravo, editors, *Ambient Assisted Living and Daily Activities - 6th International Work-Conference, IWAAL 2014, Belfast, UK, December 2-5, 2014. Proceedings*, volume 8868 of *Lecture Notes in Computer Science*, pages 91–98. Springer, 2014.

[94] Oresti Baños, Claudia Villalonga, Rafael García, Alejandro Saez, Miguel Damas, Juan Holgado-Terriza, Sungyong Lee, Hector Pomares, and Ignacio Rojas. Design, implementation and validation of a novel open framework for agile development of mobile health applications. *BioMedical Engineering OnLine*, 14:S6, 08 2015.

[95] Satya P. Singh, Aimé Lay-Ekuakille, Deepak Gangwar, Madan Kumar Sharma, and Sukrit Gupta. Deep convlstm with self-attention for human activity decoding using wearables, 2020.

[96] Mi Zhang and Alexander A. Sawchuk. USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In Anind K. Dey, Hao-Hua Chu, and Gillian R. Hayes, editors, *The 2012 ACM Conference on Ubiquitous Computing, Ubicomp '12, Pittsburgh, PA, USA, September 5-8, 2012*, pages 1036–1043. ACM, 2012.

[97] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing, ICIP 2015, Quebec City, QC, Canada, September 27-30, 2015*, pages 168–172. IEEE, 2015.

[98] Barbara Bruno, Fulvio Mastrogiovanni, and Antonio Sgorbissa. Wearable inertial sensors: Applications, challenges, and public test benches. *IEEE Robot. Automat. Mag.*, 22(3):116–124, 2015.

[99] Olivier Grisel, Andreas Mueller, Lars, Alexandre Gramfort, Gilles Louppe, Peter Prettenhofer, Mathieu Blondel, Vlad Niculae, Joel Nothman, Arnaud Joly, Jake Vanderplas, manoj kumar, Hanmin Qin, Thomas J Fan, Nelle Varoquaux, Robert Layton, Loïc Estève, Jan Hendrik Metzen, Nicolas Hug, Noel Dawe, Guillaume Lemaitre, Adrin Jalali, Rajagopalan (Venkat) Raghav, Johannes Schönberger, Roman Yurchak, Wei Li, Clay Woolam, Kemal Eren, Tom Dupré la Tour, and Eustache. scikit-learn/scikit-learn: scikit-learn 0.23.0, May 2020.

[100] Jessica Sena and William Robson Schwartz. Human activity recognition based on wearable sensors using multiscale dcnn ensemble. In *Anais Estendidos da XXXII Conference on Graphics, Patterns and Images*, pages 112–118. SBC, 2019.

[101] Michael Waskom, Olga Botvinnik, Joel Ostblom, Maoz Gelbart, Saulius Lukauskas, Paul Hobson, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Corban Swain, Alistair Miles, Thomas Brunner, Drew O'Kane, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, and Brian. mwaskom/seaborn: v0.10.1 (april 2020), April 2020.

[102] Hyun-Jun Kim, Mira Kim, Sun-Jar Lee, and Young S. Choi. An analysis of eating activities for automatic food type recognition. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–5, 2012.

[103] Netron - a viewer for neural network, deep learning and machine learning models. `https://github.com/lutzroeder/Netron`. Accessed: 2020-05-15.

[104] Yuqing Chen and Yang Xue. A deep learning approach to human activity recognition based on single accelerometer. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1488–1492, 2015.

[105] Cagatay Catal, Selin Tufekci, Elif Pirmit, and Guner Kocabag. On the use of ensemble of classifiers for accelerometer-based activity recognition. *Applied Soft Computing*, 46, 01 2015.

[106] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM*

International Conference on Multimedia, MM '15, page 1307–1310, New York, NY, USA, 2015. Association for Computing Machinery.

[107] Sojeong Ha, Jeong-Min Yun, and Seungjin Choi. Multi-modal convolutional neural networks for activity recognition. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3017–3022, 2015.

[108] Sojeong Ha and Seungjin Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 381–388, 2016.

Appendix A

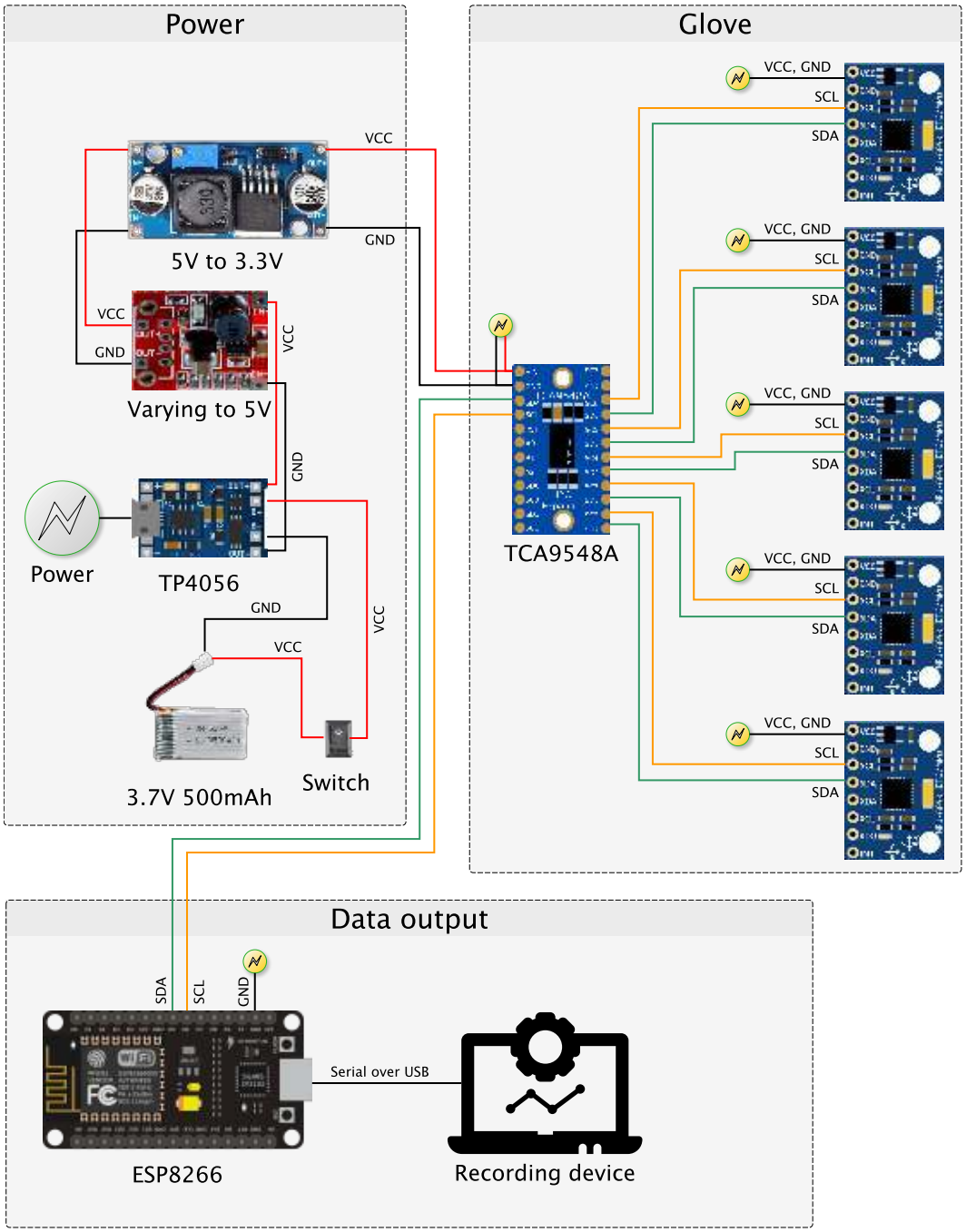


Figure A.1: WaveGlove wiring diagram

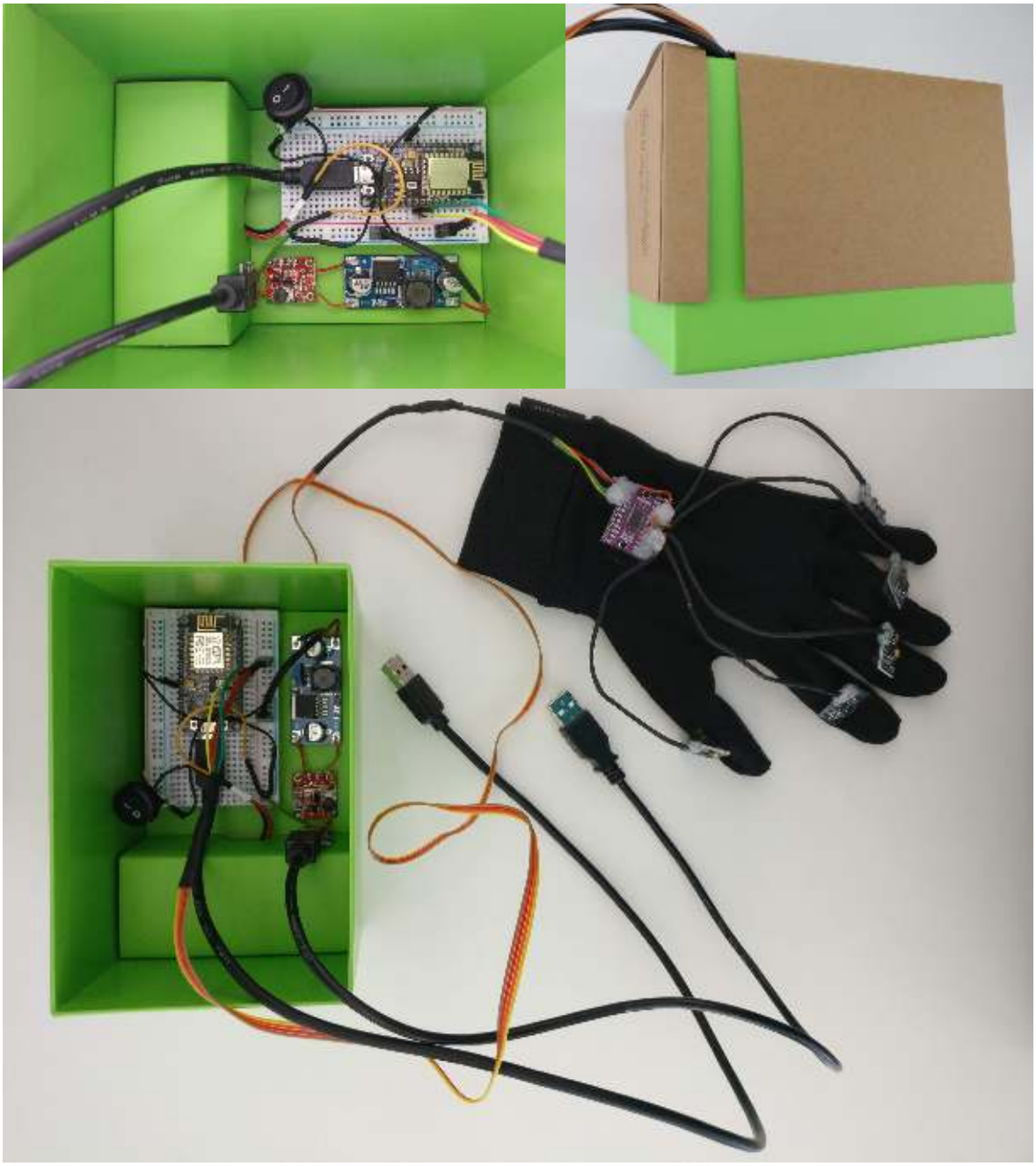


Figure A.2: A sufficiently long cable separates the glove and the rest of the wiring. Components are enclosed in a carton box with a removable lid, offering protection from the environment.

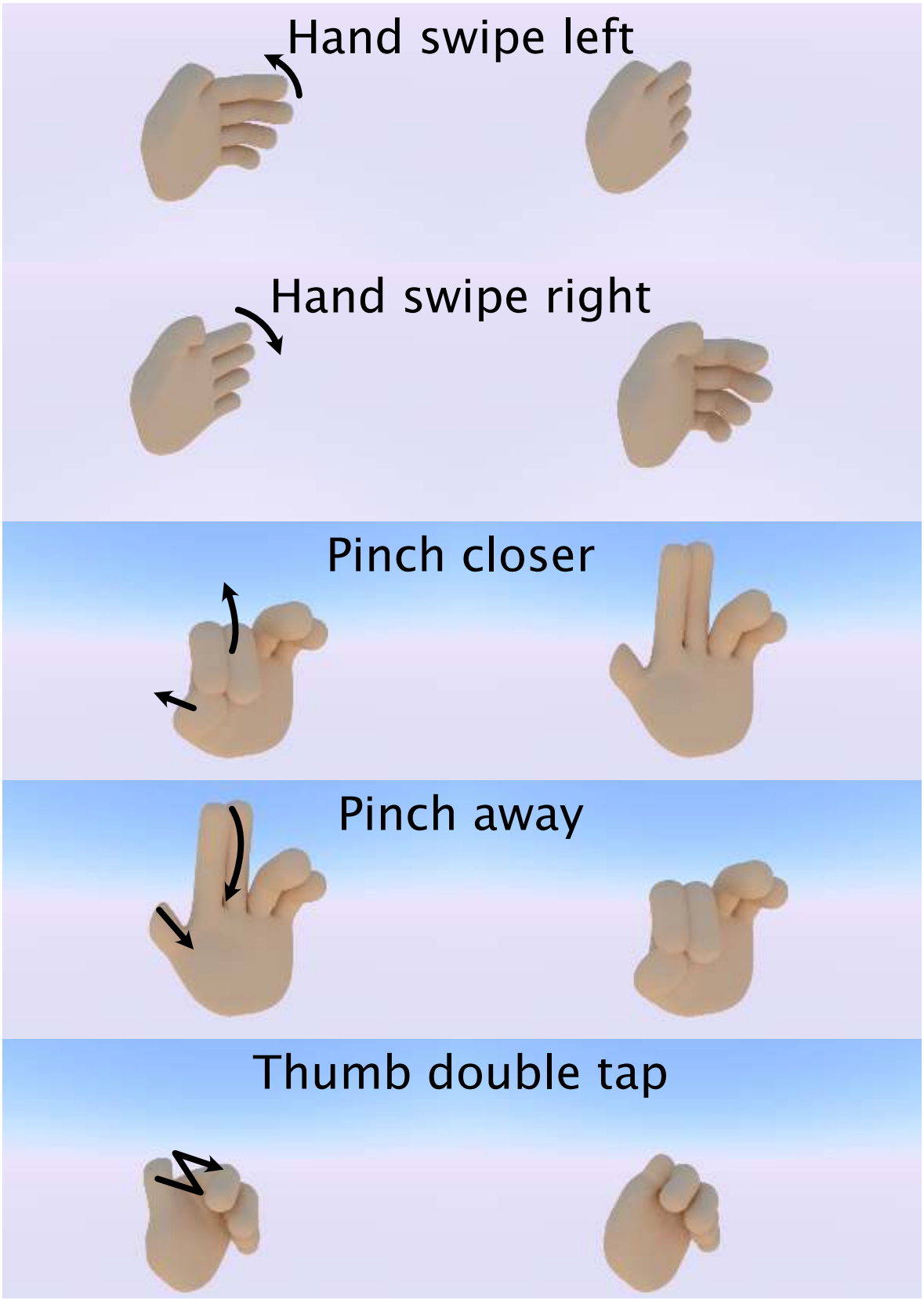


Figure A.3: First five of the gestures in the WaveGlove-multi gesture vocabulary (the arrows are used for clarification and are not present during experiments).

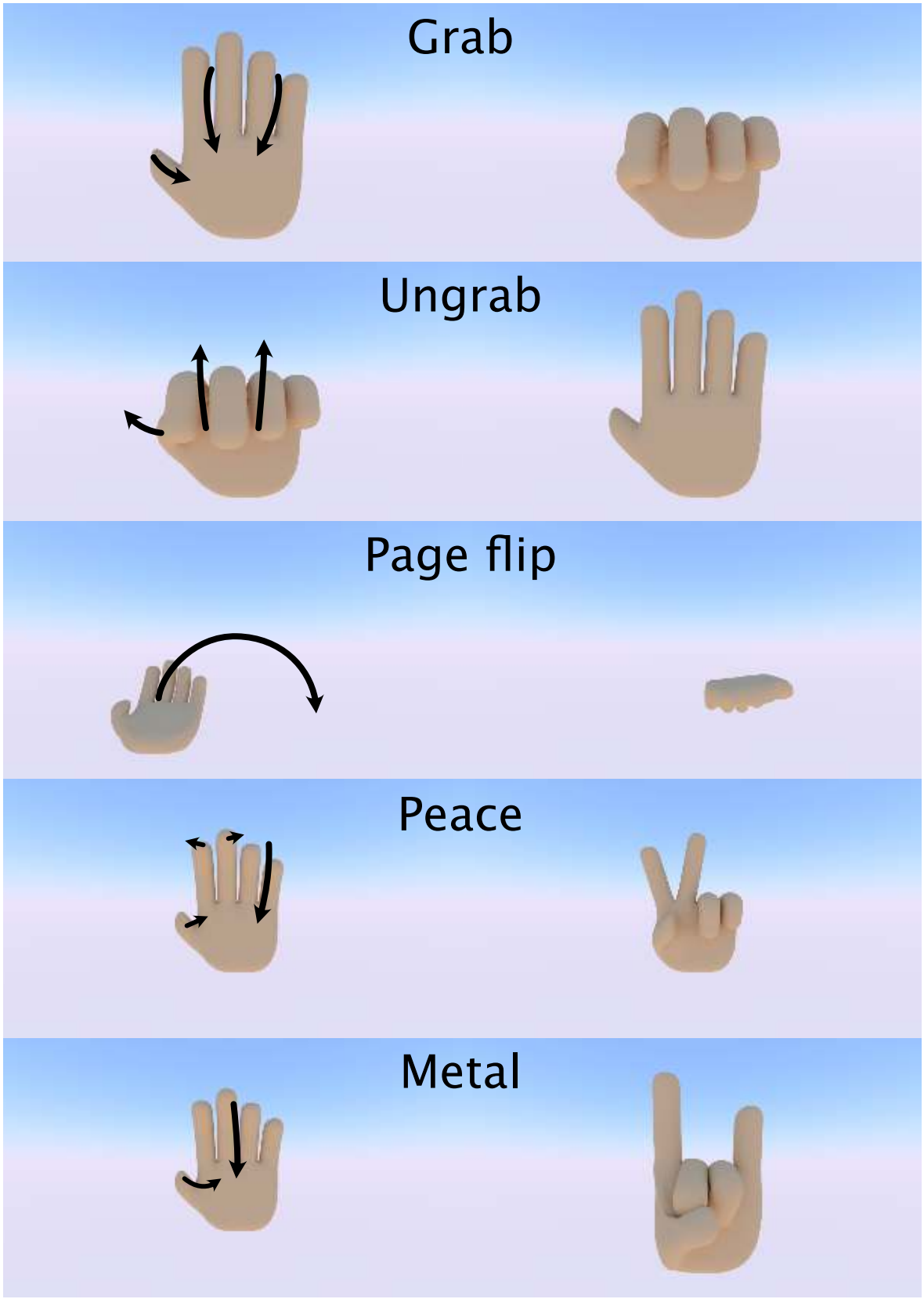


Figure A.4: The other half of the gestures in the WaveGlove-multi gesture vocabulary (the arrows are used for clarification and are not present during experiments).

Figure A.5: First slides from the tutorial recording session. In the top left, we ask the subject for a name and provide basic slide advancement instructions. Top right slides provides further instructions on how the session will progress.
The slide at the bottom serves as as the first contact of a subject with the actual recording - all the controls are explained and a chance to freely test them is provided. During recording, the live signal is shown (upper part of the window - accelerations and angular velocities below them), an animation or picture instructing on the gesture, as well as the already recorded instances and their downsampled plots. This helps the subject validate whether they did record the gesture correctly.
Also note the small progressbar at the top of the window showing the overall progress throughout a recording session.

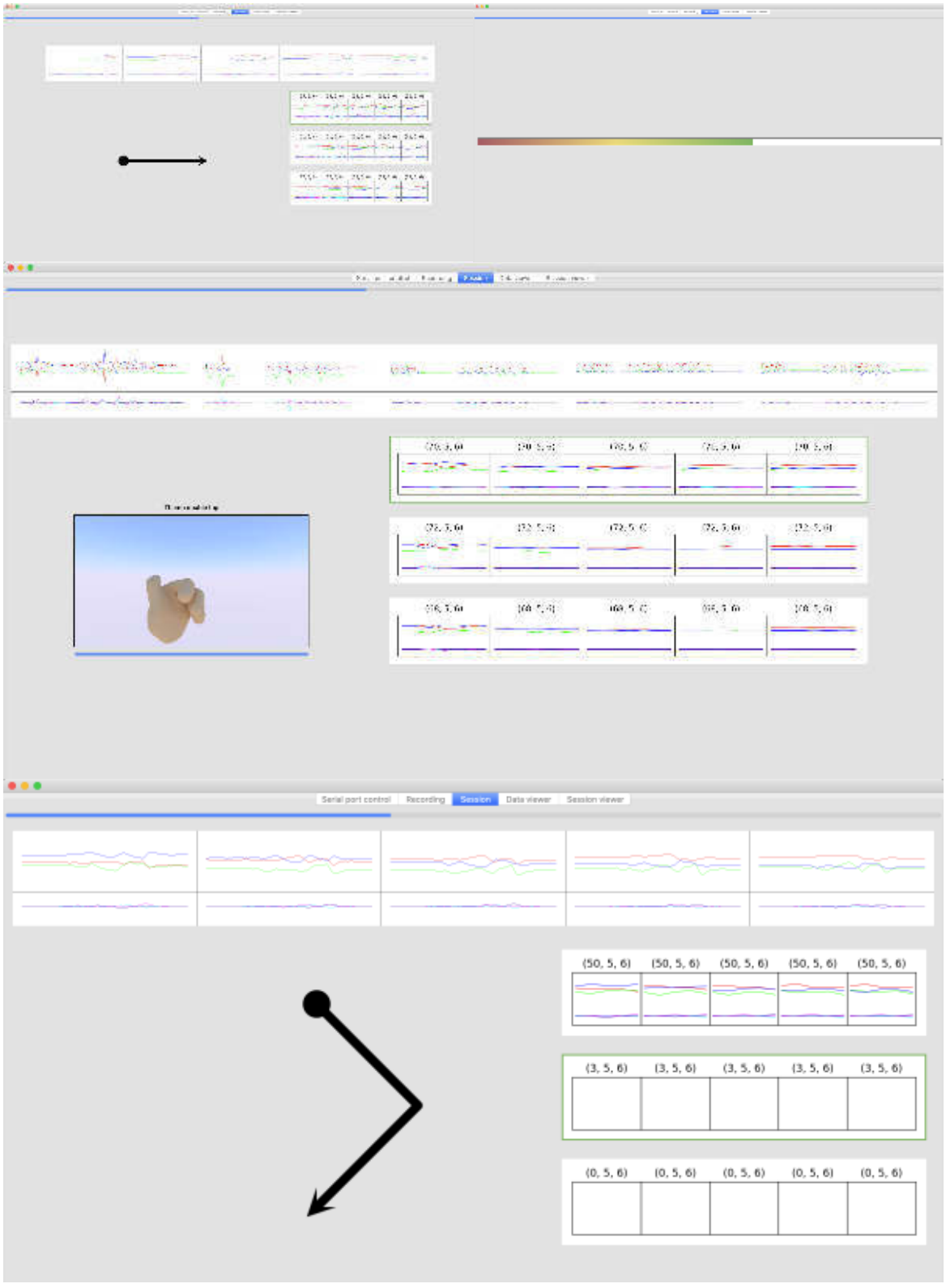
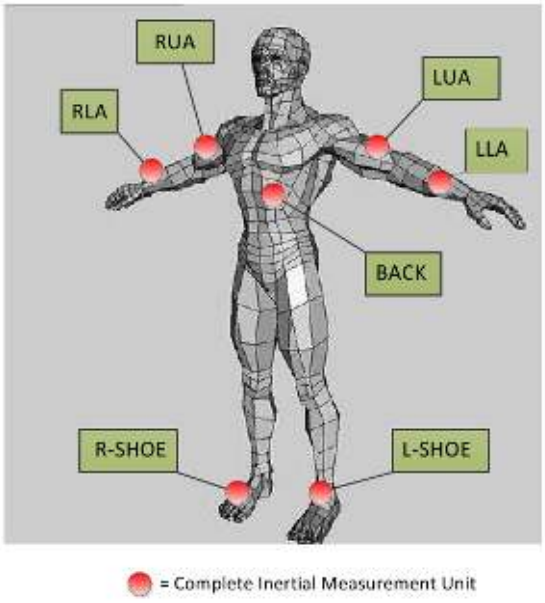
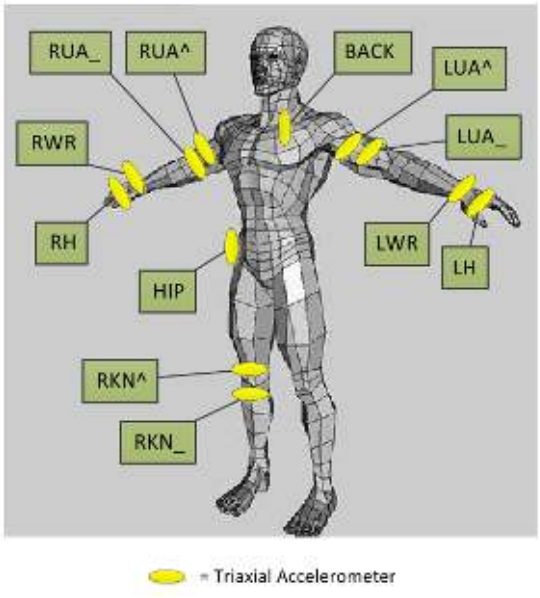


Figure A.6: Top two windows show recording a gesture (Right) from WaveGlove-single and the progressbar that is present between recording various gesture classes. In the middle a Thumb double tap gesture is being recorded. At the bottom, the subject mistakenly pressed space twice and recorded erroneous gestures of length 3 and 0. They are able to reselect the boxes (green border) and re-perform the gesture.

Appendix B

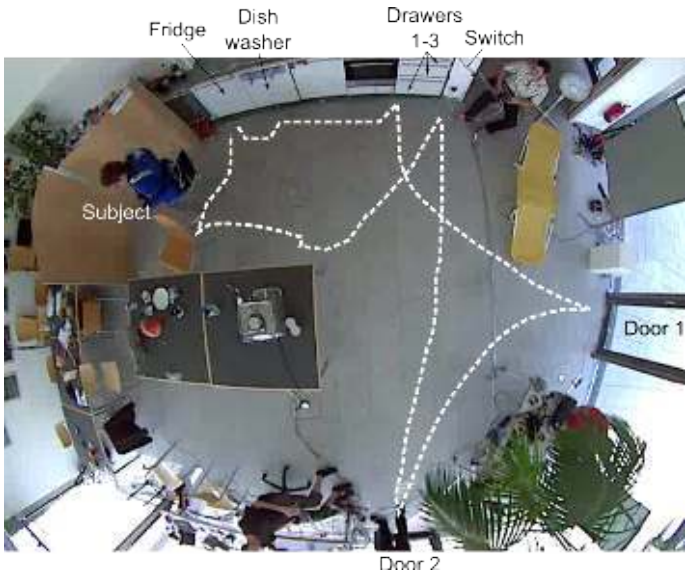


(a) Locations of the seven IMUs.



(b) Locations of the twelve accelerometers.

(c) Custom jacket housing the sensors.

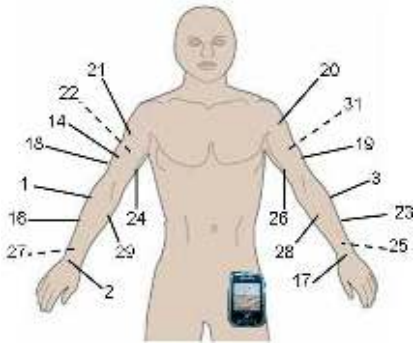


(d) Top-down view of the environment.

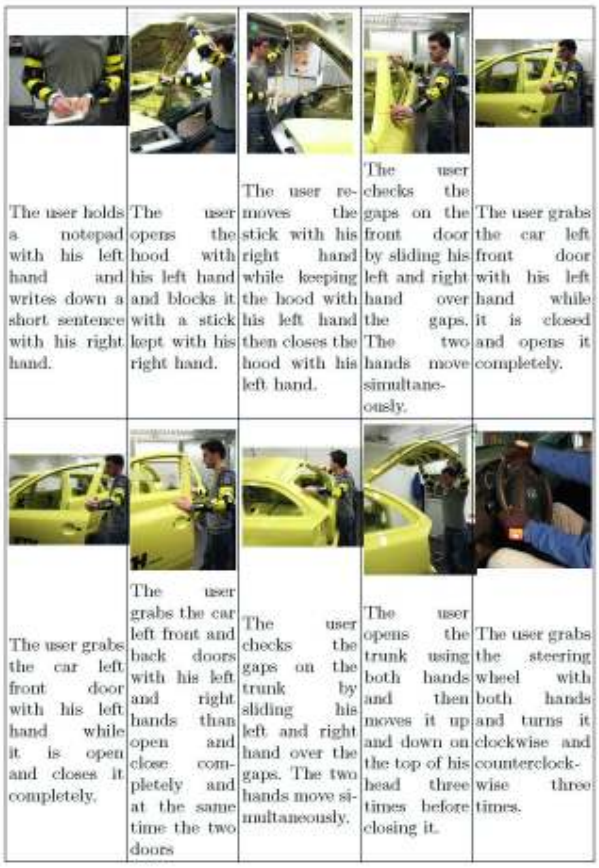
Figure B.1: Further information related to the creation of the OPPORTUNITY dataset.

	subject101	subject102	subject103	subject104	subject105	subject106	subject107	subject108	subject109	Sum	Nr. of subjects
1 – lying	271.86	234.29	220.43	230.46	236.98	233.39	256.1	241.64	0	1925.15	8
2 – sitting	234.79	223.44	287.6	254.91	268.63	230.4	122.81	229.22	0	1851.8	8
3 – standing	217.16	255.75	205.32	247.05	221.31	243.55	257.5	251.59	0	1899.23	8
4 – walking	222.52	325.32	290.35	319.31	320.32	257.2	337.19	315.32	0	2387.53	8
5 – running	212.64	92.37	0	0	246.45	228.24	36.91	165.31	0	981.92	6
6 – cycling	235.74	251.07	0	226.98	245.76	204.85	226.79	254.74	0	1645.93	7
7 – Nordic walking	202.64	297.38	0	275.32	262.7	266.85	287.24	288.87	0	1881	7
9 – watching TV	836.45	0	0	0	0	0	0	0	0	836.45	1
10 – computer work	0	0	0	0	1108.82	617.76	0	687.24	685.49	3099.31	4
11 – car driving	545.18	0	0	0	0	0	0	0	0	545.18	1
12 – ascending stairs	158.88	173.4	103.87	166.92	142.79	132.89	176.44	116.81	0	1172	8
13 – descending stairs	148.97	152.11	152.72	142.83	127.25	112.7	116.16	96.53	0	1049.27	8
16 – vacuum cleaning	229.4	206.82	203.24	200.36	244.44	210.77	215.51	242.91	0	1753.45	8
17 – ironing	235.72	288.79	279.74	249.94	330.33	377.43	294.98	329.89	0	2386.82	8
18 – folding laundry	271.13	0	0	0	0	217.85	0	236.49	273.27	998.74	4
19 – house cleaning	540.88	0	0	0	284.87	287.13	0	416.9	342.05	1871.83	5
20 – playing soccer	0	0	0	0	0	0	0	181.24	287.88	469.12	2
24 – rope jumping	129.11	132.61	0	0	77.32	2.55	0	88.05	63.9	493.54	6
Labeled total	4693.07	2633.35	1743.27	2314.08	4117.97	3623.56	2327.63	4142.75	1652.59	27248.27	
Total	6957.67	4469.99	2528.32	3295.75	5295.54	4917.78	3135.98	5884.41	2019.47	38504.91	

Table B.1: Distribution of activities among test subjects in the PAMAP2 dataset (values shown in seconds).



(a) Locations of the 20 sensors on the arms and body.



(b) Desription of the ten car maintenance activities performed.

Figure B.2: Skoda Mini Checkpoint dataset information. Ten sensors were placed around the subjects right arm, nine sensors were placed on the left arm and one sensor was placed on the left hip.

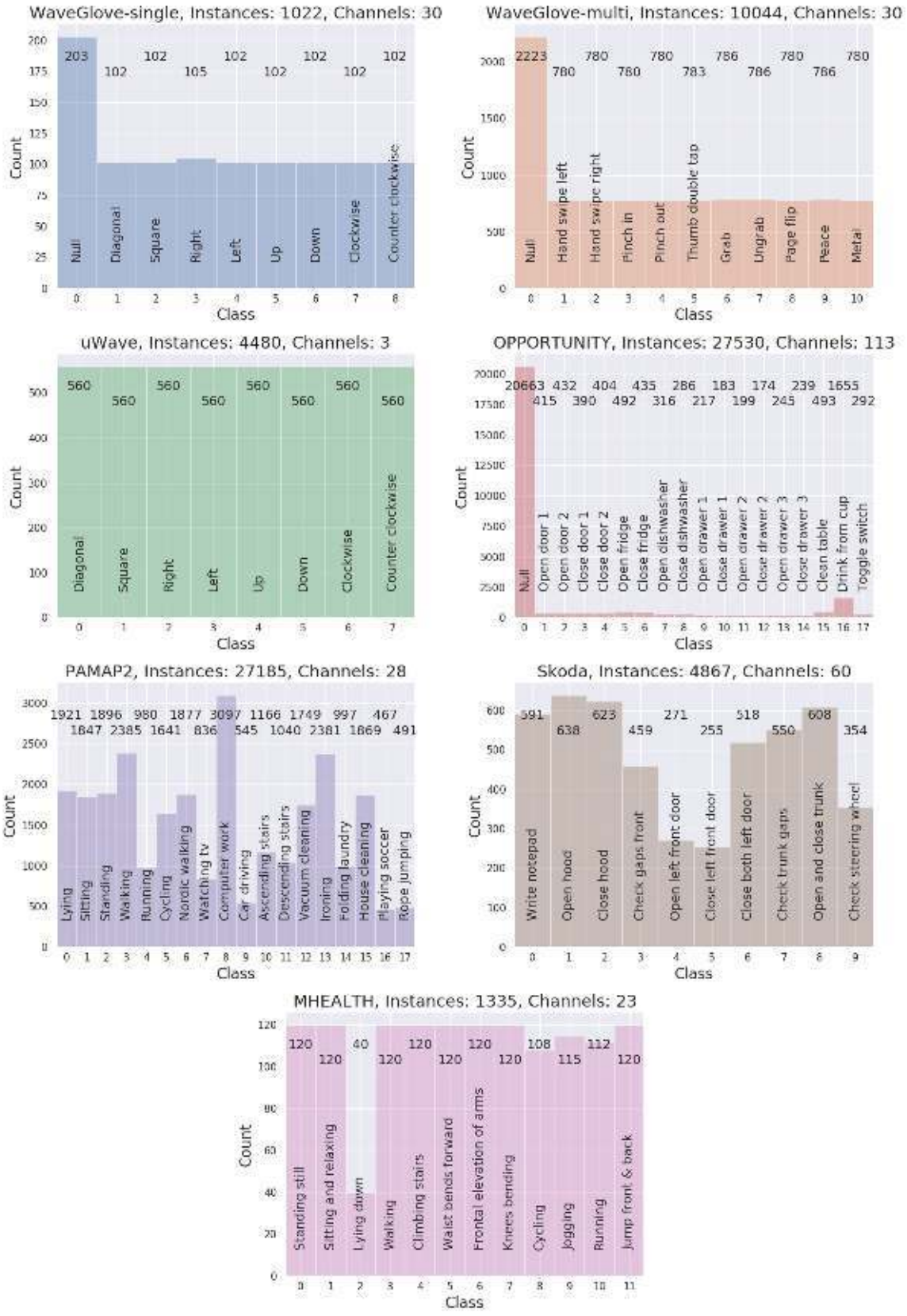


Figure B.3: Class distribution of the used datasets.

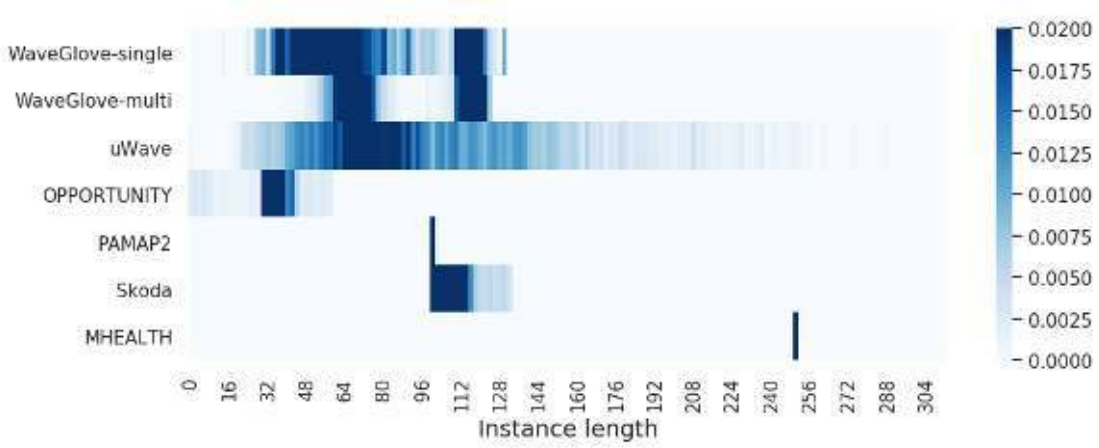


Figure B.4: Length distribution of the segmented samples.

Appendix C

The acquired datasets, the WaveGlove firmware and the code of the recording framework can be found on the attached storage medium.

Alternatively, these resources are published online:

- The acquired and used datasets are available at:
  `https://zenodo.org/record/3831958`
- The recording framework is available at:
  `https://github.com/Zajozor/gesture`
- The implementation of the models available at:
  `https://github.com/Zajozor/waveglove`