
Department of Cellular, Computational and Integrative Biology

Master's Degree in
Quantitative and Computational Biology

Final Dissertation

# Analysis of Wearable Sensors Data for Biomedical Applications

Supervisor

Prof. Mario Lauria

Student

Gabriele Berrera

Co-Supervisor

Prof. Andrea Passerini

Academic year 2020/2021

# Contents

# Introduction

In recent years, with the enhancement of technology, multiple wearable devices started making their way into our daily life. Smartphones and smartwatches are nowadays widely diffused globally, and new useful functionalities are constantly under development. Different embedded sensors are now integrated into almost all devices: from the accelerometer and gyroscope, recording motion data, to the more recent sensors, with the ability to detect Electrocardiogram (ECG) and blood pressure signals.

Along with the increase in popularity of these devices came a huge interest in researching effective methods to analyse the collected sensors data. Indeed, being able to automatically interpret the extracted signals can lead to a variety of practical applications. The monitoring of some physiological parameters and the detection of the motion activity is, for example, two of the main features already implemented in nowadays smart devices. Soon, the analysis of wearable sensors data may have the potential to become a powerful tool in the biomedical field. Indeed, this type of non-invasive technology could provide a multimodal profile of the patients' status: detecting early stages of diseases and pathologies and directly supporting professional diagnosis.

In this work we focused on the analysis of sequential data provided by wearable sensors, testing different machine learning models, specifically built to deal with the time sequences classification problem. The obtained results provided a general overview of the most effective methods, and the best performing class of algorithms was tested on two hypothetical biomedical applications: the early diagnosis of Parkinson's Disease; and the classification of Electrocardiogram (ECG) signals for diagnostical purposes.

The pipeline of this work can be divided into two main phases:

**First Phase** Firstly, a well-known sequence classification problem was chosen in order to become familiarized with the data, facilitated by the support of a solid literal background for comparing the results. The chosen task was the Human Activity Recognition (HAR), in which the aim is recognizing the activity that an individual is performing, by analysing the movement data extracted from wearable sensors. Multiple classification models were evaluated on this problem, and the results were compared to establish the most promising ones.

The first tool to be applied was SCUDO, a rank-based method developed by the COSBI institute. This algorithm was mainly designed for the analysis of gene expression profiles, but it can potentially be applied to a variety of data types. Since it was never tested on a time sequence classification task, we were interested in evaluating its performance. The next algorithm to be applied was a Support Vector Machine (SVM), aiming to reproduce the results provided in [14].

Since both SCUDO and SVM are not designed to work directly with sequential data, a set of representative features for each sequence needed to be computed. This feature extraction process transformed the original problem into a "traditional" classification task, with the drawback of losing the intrinsic time-dependent information encoded in the data. To overcome this issue, a sequence-specific classification method was tested, called HHME and described in [7]. This solution is composed of an ensemble of Hidden Markov Models, computing probability estimates for each sequence, and applying a final classification algorithm to perform predictions. Some modifications were then applied to the original architecture of the proposed model, significantly enhancing its classification power.

The final set of algorithms tested in this preliminary analysis was the Artificial Neural Networks (ANN). The Deep Learning approach is becoming very popular in the last few years for the analysis of complex structured data, such as images and sequences.

In detail, we put the focus on the family of Convolutional Neural Networks (CNN), a specific type of ANN that applies the concept of convolution to autonomously learn informative features from the raw data.

Neural Networks (CNN), a specific type of ANN that applies the concept of convolution to autonomously learn informative features from the raw data. Other types of ANN could also be used for this purpose: recurrent Neural Networks are specifically designed to model sequential data, but in most applications, when the training set is sufficiently large, CNN-based architectures resulted in a better performance [22][17][16].

The comparison of the results obtained by the different algorithms on the HAR task revealed the CNN solution to be the best performing model, even though both SVM and the improved version of the HMM-Ensemble model obtained good classification performances. Particularly, a specific deep CNN architecture, called 1D-ResNet, was able to outperform the other methods, obtaining a higher accuracy on the testing set.

**Second Phase** Once the preliminary analysis on the HAR task was concluded, we proceeded in testing the classification performance of CNN-based models on the two previously mentioned biomedical applications.

The first considered task was the early detection of Parkinson's Disease. This progressive neurodegenerative condition is associated with significant disability, with characteristic clinical manifestations including difficulty with coordinated movement such as asymmetric resting tremor, rigidity, and bradykinesia [11]. In 2017, Sage Bionetworks started a *Dream Challenge* [16] intending to research new methods that can extract digital biomarkers reflective of Parkinson's Disease. The evaluated methods were trained on motion data provided by the smartphone embedded sensors. Once we obtained the permissions to access the data collected for the challenge, we evaluated the performance of CNN models on the Parkinson's diagnosis task. As a starting point, we replicated the solution proposed by the winners of the competition [25], in which an ensemble of CNNs was used to perform the classification of the acceleration and angular velocity sequences. Other experiments were subsequently performed: modifying the original architecture and applying filtering on the dataset to improve the model performance and further explore the nature of the data.

The last case of study was the ECG signals classification. Electrocardiography is a very common clinical procedure that permits monitoring heart activity through the application of electrodes on different parts of the body. The produced signal can be very informative for the detection of different types of cardiac disorders and pathologies. In 2020 the biggest publicly available ECG dataset was released on PysioNet (PBL-XL [20]), containing more than 20.000 ECG annotated signals. Different classification algorithms were already evaluated on this dataset, as described in [17]. Our Deep Learning models were therefore tested to detect 4 different cardiac disorders: Conduction Disturbance, Myocardial Infarction, Hypertrophy and ST/T change.

This thesis is structured as follows:

- **Chapter 1:** Materials and methods. This section describes the details about the dataset employed in our experiments: the UCI HAR dataset for the Human Activity Recognition, the mPower dataset for Parkinson's detection, and the PTB-XL dataset for the ECG classification. Subsequently, a brief description of the principal involved machine learning baselines was made, highlighting some of the foremost necessary concepts that are key to understanding their functioning, and providing some literature in merit.
- **Chapter 2:** Results. A detailed description of the experiments is given in this chapter, putting the focus on the chosen architectures and their implementation. All the results produced by the tested algorithms are reported and quantified through different evaluation metrics to compare their performance.
- **Chapter 3:** Discussion. The results obtained during the different experiments were then critically discussed, reflecting on their possible implications and making a comparison with the literature.

- **Chapter 4:** Conclusions. This chapter contains a summary of the key points discussed in this thesis, including some hypotheses concerning future work and prospects.

# 1 Materials and Methods

## 1.1 Datasets

The time sequences classification methods were trained and evaluated on three different datasets, representing the three cases of study. The UCI HAR public dataset, the mPower dataset, and the PTB-XL ECG dataset were respectively employed for the tasks of Human Activity Recognition, Parkinson's Disease detection and ECG diagnostic classification.

### 1.1.1 UCI HAR Dataset

For the task of Human Activity Recognition was involved a public dataset, provided by the Machine Learning Repository of the University of California, Irvine. The UCI HAR dataset[14] collects data from a group of 30 volunteers within an age range between 19-48. During data collection each individual performed six different daily life activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) while wearing a Samsung Galaxy S II. 3-axial linear acceleration and 3-axial angular velocity were sampled at a constant rate of 50Hz, utilizing the embedded accelerometer and gyroscope of the smartphone.

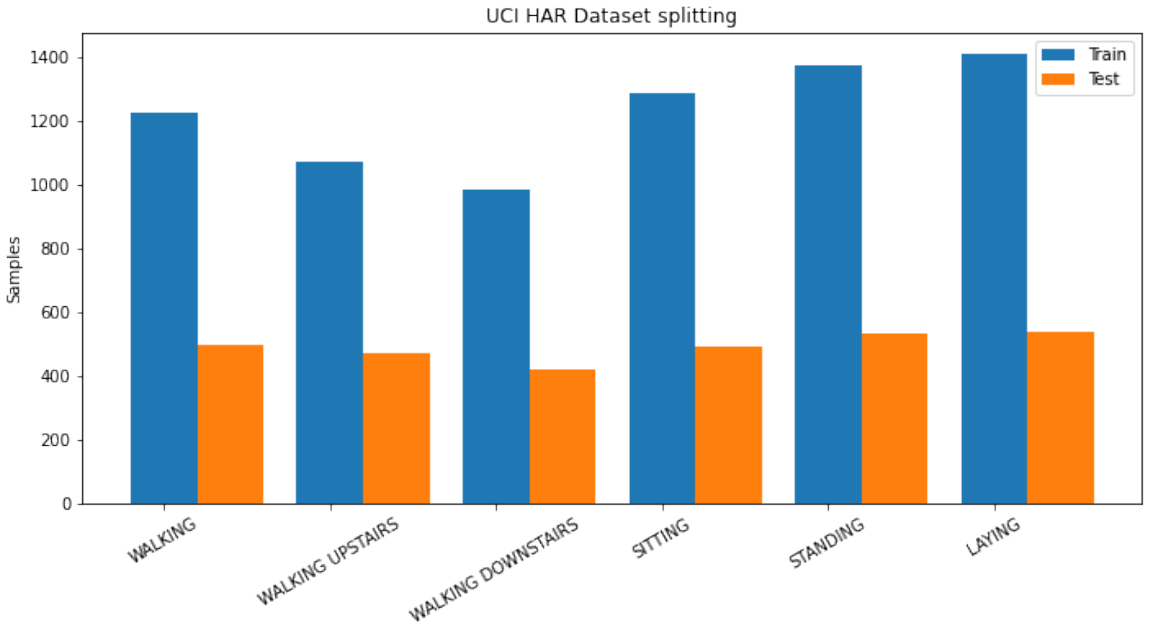


Figure 1.1: Class compositions of the UCI HAR training and testing set

The authors pre-processed then the sensor signals by applying a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz, to remove noise. Subsequently, they were sampled in fixed width sliding windows of 2.56 sec and 50% overlap (128 readings per window). The sensor acceleration signal, which had gravitational and body motion components, was separated into body acceleration and gravity using a Butterworth low-pass filter with a corner frequency of 0.3 Hz. Finally, a Fast Fourier Transform (FFT) was applied to some of the signals, and magnitude was calculated using the Euclidean norm.

From the raw filtered data, Reyes et. al. computed a set of 561 different representative features for each recorded sequence. All the signals used to estimate the variables are described in Table 1.1.

| Signal Name | Description |
|---|---|
| *tBodyAcc-XYZ* | triaxial acceleration signal without gravity |
| *tGravityAcc-XYZ* | triaxial acceleration gravity signal |
| *tBodyAccJerk-XYZ* | triaxial body acceleration signal derived in time |
| *tBodyGyro-XYZ* | triaxial angular velocity signal |
| *tBodyGyroJerk-XYZ* | triaxial angular velocity signal derived in time |
| *tGravityAccMag* | magnitude of *tGravityAcc* signal |
| *tBodyAccJerkMag* | magnitude of *tBodyAccJerk* signal |
| *tBodyGyroMag* | magnitude of *tBodyGyro* signal |
| *tBodyGyroJerkMag* | magnitude of *tBodyGyroJerk* signal |
| *fBodyAcc-XYZ* | *tBodyAcc* in the frequency domain after applying a FFT |
| *fBodyAccJerk-XYZ* | *tBodyAccJerk* in the frequency domain after applying a FFT |
| *fBodyGyro-XYZ* | *tBodyGyro* in the frequency domain after applying a FFT |
| *fBodyAccMag* | *tBodyGyroMag* in the frequency domain after applying a FFT |
| *fBodyAccJerkMag* | *tBodyAccJerkMag* in the frequency domain after applying a FFT |
| *fBodyGyroMag* | *tBodyGyroMag* in the frequency domain after applying a FFT |
| *fBodyGyroJerkMag* | *tBodyGyroJerkMag* in the frequency domain after applying a FFT |

Table 1.1: Set of signals computed from the original sensors data

For each one of these extracted signals a set of variables was then estimated in order to represent their main characteristics. In details, the computed variables are reported in Table 1.2

| Variable Name | Description |
|---|---|
| *mean()* | Mean value |
| *std()* | Standard deviation |
| *mad()* | Median absolute deviation |
| *max()* | Largest value in array |
| *min()* | Smallest value in array |
| *sma()* | Signal magnitude area |
| *energy()* | Energy measure. Sum of the squares divided by the number of values. |
| *iqr()* | Interquartile range |
| *entropy()* | Signal entropy |
| *arCoeff()* | Autorregresion coefficients with Burg order equal to 4 |
| *correlation()* | correlation coefficient between two signals |
| *maxInds()* | index of the frequency component with largest magnitude |
| *meanFreq()* | Weighted average of the frequency components to obtain a mean frequency |
| *skewness()* | skewness of the frequency domain signal |
| *kurtosis()* | kurtosis of the frequency domain signal |
| *bandsEnergy()* | Energy of a frequency interval within the 64 bins of the FFT of each window. |
| *angle()* | Angle between to vectors. |

Table 1.2: Set of variables evaluated for each of the computed signals.

Finally, the obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data. The composition of those two sets is shown in the Figure 1.1

### 1.1.2 mPower dataset

For the experiments on the Parkinson's Disease detection, data were provided by Parkinson's Disease Digital Biomarker DREAM Challenge [16]. The goal of this challenge was to benchmark methods for processing sensor data and developing digital signatures that can be reflective of the disease. In particular, we focused on the first sub challenge in which the mPower study dataset was used to perform classification between Parkinson's patients and control subjects.

The mPower dataset [3] consists of data collected from an iPhone application, developed by Sage Bionetworks. The application recorded data from self-reported patients and controls, daily administering them four separate activities: 'memory' (a memory-based matching game), 'tapping' (measuring the dexterity and speed of two-finger tapping), 'voice' (measuring sustained phonation by recording a 10-s sustained 'Aaaahhh'), and 'walking' (measuring participants' gait and balance via the phone's accelerometer and gyroscope). For our experiments we focused only on the 'walking' sequences, since we were mainly interested in analysing sensors data, that could potentially be collected in a passive manner.

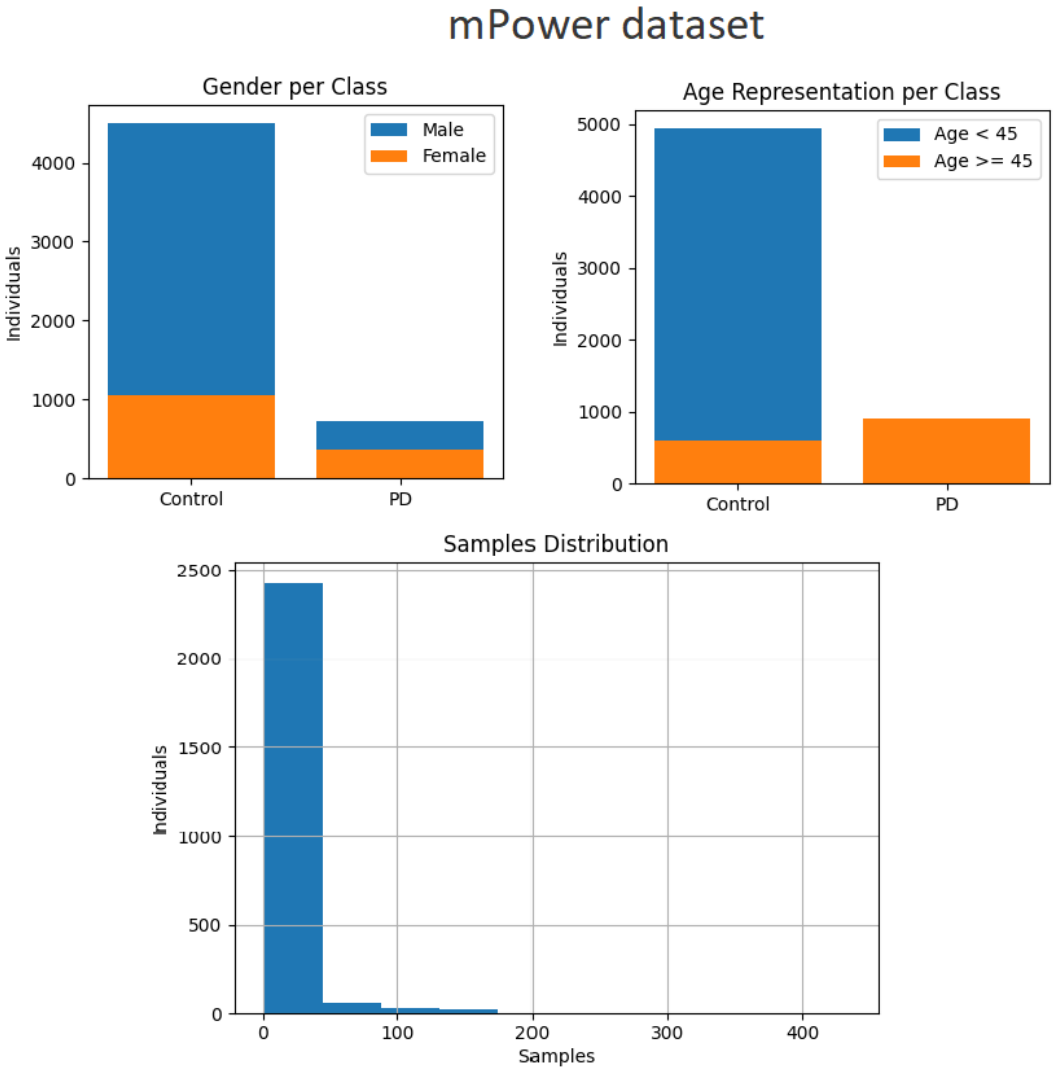


Figure 1.2: mPower dataset structure in terms of gender, age and number of recorded sequences per each individual.

The dataset contains records from a total of 2804 subjects, 656 self-reported Prkinson's patients and 2148 healthy controls. Its general structure is described in Figure 1.2. During the walking test the mPower application asked the users to perform a simple walking task: each subject had to walk in one direction ("outbound") for 20-30 seconds, then quite stand ("rest") for another 20-30 seconds and then walk back to the starting point ("return") for the same amount of time. During this test the 3-axial acceleration and rotation signals were extracted from the accelerometer and the gyroscope, with a frequency of 100 Hz. Each subject performed the test multiple times, populating the dataset with a total of 34,632 walking records.

The dataset was downloaded from the Synapse portal using the Python APIs. In addition to the walking records, also a demographic table was present, reporting personal information about the subjects (e. g. gender, age, medical diagnosis, medical usage.)

### 1.1.3 PTB-XL ECG dataset

An Electrocardiogram (ECG) is a signal describing heart activity, and it is produced by recording the electrical voltage over time, measured by some electrodes applied on the subject's skin.

The PTB-XL ECG dataset is a large dataset of 21837 clinical 12-lead ECGs from 18885 patients[20]. 10 seconds signals were sampled at frequencies of 500Hz and 100Hz, and they were annotated according to the SCP-ECG standard[1], with 71 different statements covering diagnostic, form, and rhythm. ECG sampled signals are stored in the WaveForm DataBase (WFDB) format as proposed by the PhysioNet platform.

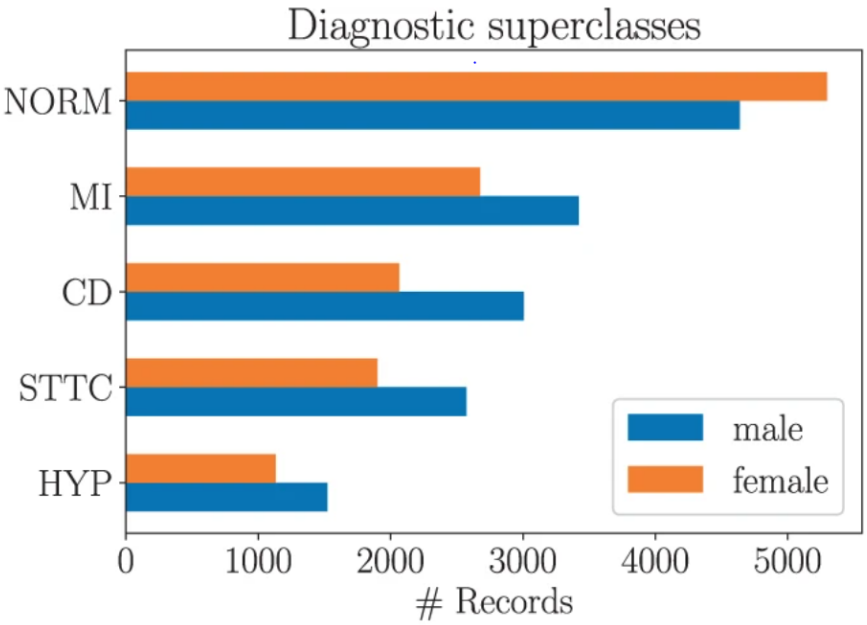


Figure 1.3: Number of samples for each of the 5 superclasses (figure from [20])

The dataset is balanced by sex, and it covers all the ages range from 0 to 95 years. For diagnostic purpose it provides a hierarchical organization of labels in terms of 5 superclasses and 24 subclassess. For our experiments we focused only on the classification of signals in the main 5 superclasses, indicating Normal ECG signal, Conduction Disturbance, Myocardial Infarction, Hypertrophy and ST/T change, respectively represented with the acronyms NORM, CD, MI, HYP and STTC. The distribution of records per class is shown in Figure 1.3. Those 5 classes are not mutually exclusive, meaning that records can represent ECG signals of patients with co-occurring diseases.

## 1.2 Classification methods

Different machine learning solutions were evaluated on their ability to classify the time sequences data. In particular, SCUDO, Support Vector Mchines (SVM), Hidden Markov Models (HMM), and Condolutional Neural Networks (CNN) were tested on the Activity Recognition task, while only CNN-based models (also in combination with Random Forest, SVM and HMM) were evaluated for the diagnostic classification of Parkinson's disease and ECG records.

### 1.2.1 SCUDO

SCUDO (Signature-based ClUstering for DiagnOstic purposes) [10] is a tool developed by the COSBI institute for the analysis of gene expression profiles, for diagnostic and classification purposes. This rank-based method relies on the construction of a reference map of transcriptional signatures from each of the groups, and the classification is performed determining the relative map position of an individual's signature[9].

The pipeline of the algorithm is well described in the Figure 1.4. As the first step a feature selection is performed using a Mann–Whitney-Wilcoxon test between profiles of the two classes in the training set. The genes removed from this step are also discarded from the testing set. Next, each of the reduced profiles are ranked by value, from the largest to the smallest, and a signature is constructed from the top *n_top* and the bottom *n_bottom* genes. A distance between each couple of signatures is then computed using a GSEA based metric, and those distances are subsequently employed to draw a map of the testing set samples. The map can be represented in a graph, with each node representing a sample, and edges of proportional length representing distances. Only the bottom $N\%$ percentile of distances is selected, in order to simplify the model and to produce a clear map.


Figure 1.4: SCUO pipline described in []

The last and crucial step is represented by the analysis of the resulting network. In some cases, the partition in clusters results very easy and can be done by hand, otherwise it is possible to use one of the many unsupervised community identification algorithms.

SCUDO is available as an online tool, however an R package version called rSCUDO was also implemented [4], and can be useful to develop more customable implementations.

Despite SCUDO was designed specifically for the classification of expression profiles, it can potentially be applied also to different types of data. In this work we tried to evaluate its performances when dealing with time series of acceleration and rotation data.

### 1.2.2 Hidden Markov Model

A Hidden Markov Model (HMM) [15] is a method that allows to fit sequence data in a relatively simple way. It is composed of two stochastic processes, one described by a Markov Chain with non-observable states (hidden), while the other is a random process that can be directly observed, producing a sequence of observations.

Let's call $X = X_1, \ldots, X_n$ the set of hidden Markov states and $Y = Y_1, \ldots, Y_m$ the set of observations. The transition probabilities from a state to another, $P(X_i|X_j)$ with $i, j = 1, \ldots, n$, are represented in a *transition matrix* while the probabilities of observing an event given a certain state, $P(Y_i|X_j)$ with $i = 1, \ldots, n$ and $j = 1, \ldots, m$, are collected in the so-called *emission matrix*.

Once this two matrices and all the stationary (or initial) state probabilities $P(X_i)$ are defined, we can compute joint probabilities in the form of $P(Y = Y_1, \ldots, Y_n, X = X1, \ldots X_n)$, and moreover it is possible to infer which is the most likely sequence of states, given a specific sequence of observations:

$$\underset{X=X_1,\ldots,X_n}{\operatorname{argmax}} \; P(X = X_1, \ldots, X_n | Y = Y_1, \ldots, Y_n)$$

Applying the Bayes Theorem we obtain:

$$\underset{X=X_1,\ldots,X_n}{\operatorname{argmax}} \frac{P(Y|X)P(X)}{P(Y)}$$

where $P(Y|X)$ represents the product $\prod_{i=1}^{n} P(Y_i|X_i)$ and $p(X) = \prod_{i=1}^{n} P(X_i|X_{i-1})$, while the term $P(Y)$ can be neglected. The resulting optimization problem can so be written in the form:

$$\underset{X=X_1,\ldots,X_n}{\operatorname{argmax}} \prod_{i=1}^{n} P(Y_i|X_i)P(X_i|X_{i-1})$$

This problem has no analytical solution, however it can be solved applying the well-known Viterbi Algorithm [5].

Another famous algorithm, the Baum-Welch[18], can be applied instead for estimating the optimal parameters of the model (stationary, transition, and emission probabilities), given a set of observation sequences. This solution implements a reestimation procedure applying the forward-backward algorithm, and it is indispensable in the case we want to fit the model for classification purpose.

### 1.2.3 Support Vector Machine

Support Vector Machines (SMV)[2] are a famous and widely used group of supervised models for classification and regression problems. In principle they were designed only to deal with linear separable problems, however thanks to the introduction of the so-called kernel trick, they became in addition a powerful tool for modelling non-linearity.

To better understand the idea behind SVM, let's consider a binary linear problem in which we have a set of n points $(x_1, y1), \ldots, (x_n, y_n)$, with $y_i$ representing the label with possible values $\{-1, 1\}$, and $x_i$ a p-dimensional real vector. Applying the so-called hard margin SVM the goal is to find the maximum distance (margin) between two hyperplanes that separates the group of points with label -1 from the group with label 1.

Assume that the hyperplane that linearly separates the two set of points is defined as $w^T x - b = 0$, with $w$ representing the normal vector to the hyperplane, and $b$ an the offset term. The margin can be defined as a couple of parallel hyperplanes $w^T x - \alpha = 0$ and $w^T x - \beta = 0$, and for simplicity let's define $\alpha = b + 1$ and $\beta = b - 1$. The two hyperplanes can now be written as:

$$w^T x – b = 1$$
$$w^T x – b = -1$$

All the data points must lay outside the margin, in other words we want that the following constraint is satisfied for each point $x_i$:

$$y_i(w^T x_i – b) \geq 1 for i = 1, .., n \tag{1.1}$$

The aim is maximizing the distance between the two hyperplanes, computed as $\frac{2}{||w||}$, that is equal to minimizing $\frac{||w||}{2}$. For simplicity the squared form is generally considered, and we obtain the following optimization problem:

$$min_{w,b} \frac{1}{2}||w||^2 = min_{w,b} \frac{1}{2} w^T w$$
$$s.t. y_i(w^T x_i - b) \geq 1$$

The minimization of the obtained convex function is called the primal problem and can be solved introducing Lagrange multipliers $\langle \alpha_i \rangle$, obtaining the corresponding dual problem, in the following form:

$$max_\alpha - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0$$

In the case our problem is not linearly separable we can apply the soft margin SVM solution. Moreover, this model can also be applied in linear problems with the aim of avoiding overfitting, often caused by the hard margin solution when the distance between the two hyperplanes is too small. The idea behind the Soft Margin SVM is allowing misclassification of points, adding a penalty computed though a loss function, proportional to the number of misclassified points. The obtained primal problem is in the form:

$$min\frac{1}{2}w^T w + C\sum_{i=1}^{n}\zeta_i$$

$$s.t. y_i(w^T x_i - b) \geq 1 - \zeta_i, \forall i = 1, .., n, \zeta_i \geq 0$$

where $C$ represents a trade-off parameter between the margin size and the number of misclassified point. As for the hard margin solution, it is possible to introduce the Lagrange multipliers to obtain the following dual problem:

$$max_\alpha - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0, 0 \leq \alpha_i \leq C$$

As mentioned before, to extend the use of SVMs in a variety of nonlinear problems it is possible to apply the kernel trick. With this method every dot product is substituted with a non-linear kernel function of choice. This function takes in input the data points in the original lower dimensional space, and returns the dot product of the vectors in a higher dimensional feature space, in which the problem is linearly solvable. The power of kernels is that they don't explicitly transform the data in the higher dimensional space, but instead they represent the transformed vectors as a function in the original lower dimensional space. Common kernel functions are Polynomial kernels (homogeneous and inhomogeneous), hyperbolic tangent and Gaussian radial basis function.

### 1.2.4 Random Forest

Random Forest[19] is a commonly used method for solving classification and regression problems. The concept behind this model is creating an ensemble of multiple Decision Trees, applying then a voting system to perform the class prediction, or averaging on the results for regression tasks.

A Decision Tree encapsulate the data in a tree structure, splitting the original set following classification features-based rules, with the goal of creating a model represented by a set of decision steps allowing to make data prediction. The construction of a Decision Tree is therefore a recursive procedure, in which the original set of data (represented as the root node) is partitioned in subsets (represented as the children nodes) following the predetermined rules, until all the elements in the new sets have all the same target value, or no more information is added with further splits. Typical metrics to establish the splitting rules are Gini Index, Entropy, and Information Gain.

The main problem with Decision Trees is that they are prone to overfitting, especially when deep structures are built. Random Forest are used to mitigate this problem, averaging the results of multiple Decision Trees.

Let's consider an original set of data $X = x_1, ..x_n$ with corresponding targets $Y = y_1, ..., y_n$, in which each element $x_i = (\alpha_1^i, ..., \alpha_m^i)$ is a vector of computed features for the i-th record. The general procedure for creating a Random Forest model, with number of trained Decision Trees equal to $T$, is composed by the following steps:
For each $t$ in $\{1, ..., T\}$:

- Sample with repetition $n$ training records from $X, Y$, and select a random subset of considered features. Let's call the new sets $X_t$ and $Y_t$, in which each element $x_i^t$ in $X_t$ is represented as a vector composed by the computed values for the considered subset of features.
- Train a Decision Tree model $f_t$ on the sets $X_t, Y_t$

The resulting ensemble of Decision Trees can finally compute prediction on testing data $z$, considering the majority output between all the trees, in the case of classification problems, or averaging the results, in regression problems, as:

$$pred = \frac{1}{T} \sum_{t=1}^{T} f_t(z)$$

### 1.2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [24] are a subset of Artificial Neural Networks (ANN) in which one or more convolutional layers are present. As every ANN they are built connecting multiple artificial neurons, or perceptrons, creating a graph structure composed by different layers.

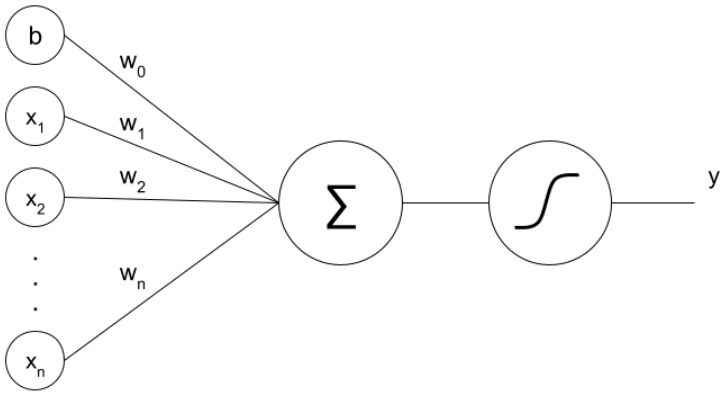


Figure 1.5: Graphical representation of a perceptron.

The graphical representation of a perceptron is shown in Figure 1.5. This simple model works by computing the product between the input vector $X$ and a vector of learned weights $W$, eventually applying a bias term $b$. Subsequently the summation of the resulting vector is computed, and a so-called activation function $\sigma$ is applied to produce the final output $y$:

$$y = \sigma(b + \sum_i x_i w_i)$$

Connecting multiple layers of artificial neurons in a fully connected way we obtain a Multi-Layer Perceptron (MLP), that can be trained optimizing the parameters of each node with the backpropagation algorithm. The goal is minimizing a non-convex Loss Function: at each training iteration (epoch) the error is computed for the final output and is then backpropagated to the internal nodes, in order to optimize all the parameters using a gradient descent-based method.

Due to their full connectivity MLP have the tendency of overfitting the data. CNN try to solve this problem using a specific network layer called convolutional layer, taking advantage of the structured nature of the data.

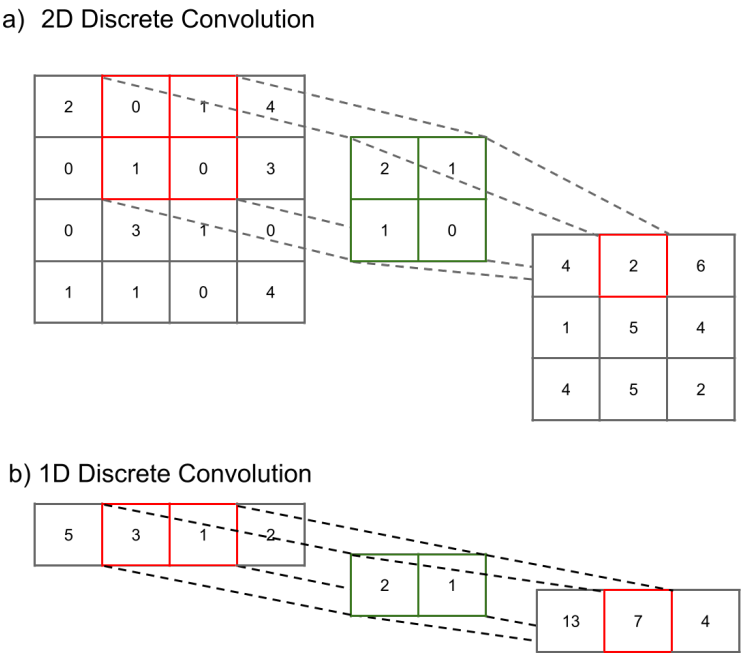


Figure 1.6: Convolution examples in: a) 2 dimensions and b) 1 dimension

Convolution is a mathematical operation involving two different functions $f$ and $g$, that produce a third "blended" function $(f * g)$ representing how the shape of one is affected by the other. It is computed as the integral of the product between $f$ and $g$, with one of the two functions shifted by an amount $t$:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

When we are in a discrete domain, like in the case of time sequences data, the discrete version of convolution can be applied. Suppose we have a time sequence, represented as a function $f$, and another discrete function $g$, both defined in the discrete set of time steps $i = [0, 1, .., n]$.

The convolution function computed between $f$ and $g$ can be defined as:

$$(f * g)[x] := \sum_{i=0}^{n} f[i]g[i - x]$$

A practical example of the convolutional process, for 1- and 2-dimensional data is shown in Figure 1.6. The $f$ function represents the input data, while $g$ represents the so-called kernel. The kernel can be seen as a typically small, fixed size sliding window of weights. The element-wise multiplication is applied between a portion of the input and the kernel, and the final output corresponds to the summation of the resulting values. The kernel is then shifted by a certain amount (called stride) to the next area of the input, repeating the procedure until all the input has been processed.

When convolution is implemented in a neural network each perceptron in the convolutional layer is connected only with a small part of the input, and the same kernel is shared between all the layer's units. This gives the advantage of drastically decreasing the number of weights, diminishing the risk of overfitting, and furthermore decreasing the training computational cost.

Typical types of layers used in convolutional networks are:

- **Pooling layer**. Performs a downsampling of the input data. Typical pooling methods are Maximum and Average Pooling, computing respectively the maximum and the average value of each fixed size portion of the input. Global Pooling could be used instead to perform downsampling considering the entire input data.

- **Batch Normalization Layer**. Performs a normalization across the mini-batch of training samples, re-centring mean and scaling by variance.
- **Dropout Layer**. This layer is used to randomly shut down units of the network with a given rate. This allows to simulate a virtual ensemble of different network architectures, regularizing the model and preventing overfitting.

In a typical CNN architecture, these layers are combined with the convolutional layers, obtaining a powerful tool for extracting optimal features from complex and structured input data.

One or multiple Dense layers (fully connected) are often applied at the end of the network, modelling the features extracted during the convolutional process, in order to solve classification and regression tasks.

# 2 Results

As previously mentioned, the first step of our work regards the analysis of the sensors data provided by the UCI HAR dataset (chapter 1.1.1). We chose the task of Human Activity Recognition because it is a well-known time sequence classification problem, with a rich literature background. This allowed us to gain familiarity with this type of data and explore the possible solutions applicable to this problem. Different algorithms were tested and evaluated in terms of classification performance; starting from more traditional solutions as SVM and HMM-based methods, and arriving to Deep Learning approaches, that represent the currently state of the art for the classification of this structured type of data.

After this preliminary overview on the problem, some of the best performing methods were applied on the two chosen biomedical problems: the Parkinson's Disease detection and the ECG signal classification.

In the following sections are described the experiments carried out during our work, showing the architecture of the tested models and the key implementation details. All the results obtained during the three cases of study are reported and compared to each other.

## 2.1 Human Activity Recognition

As previously mentioned, for our overview on the Human Activity Recognition task we relied on the UCI HAR dataset. For our experiments we considered both the sampled 3-axial signals and the set of precomputed features, accordingly to the characteristics of each tested method.

### 2.1.1 SCUDO Classification

The first method that we were curious to apply was the gene expression classification tool SCUDO[10], since it was never tested in the classification of time sequences.

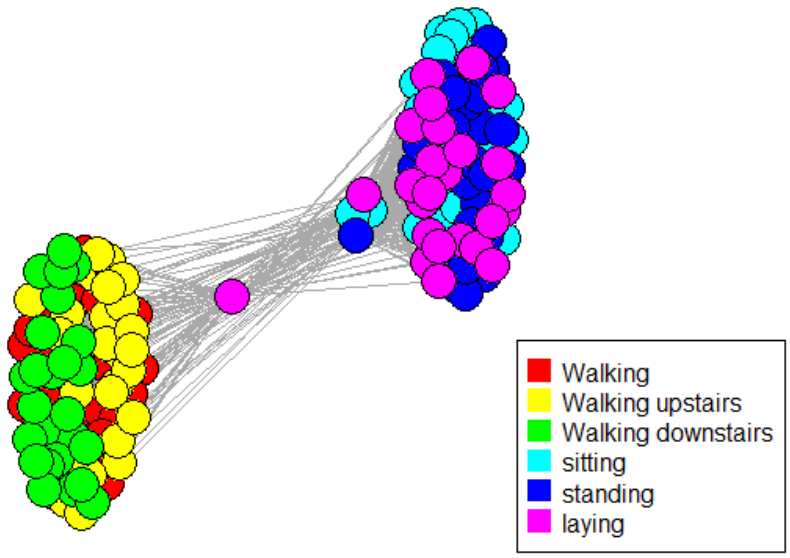


Figure 2.1: Training graph computed by SCUDO

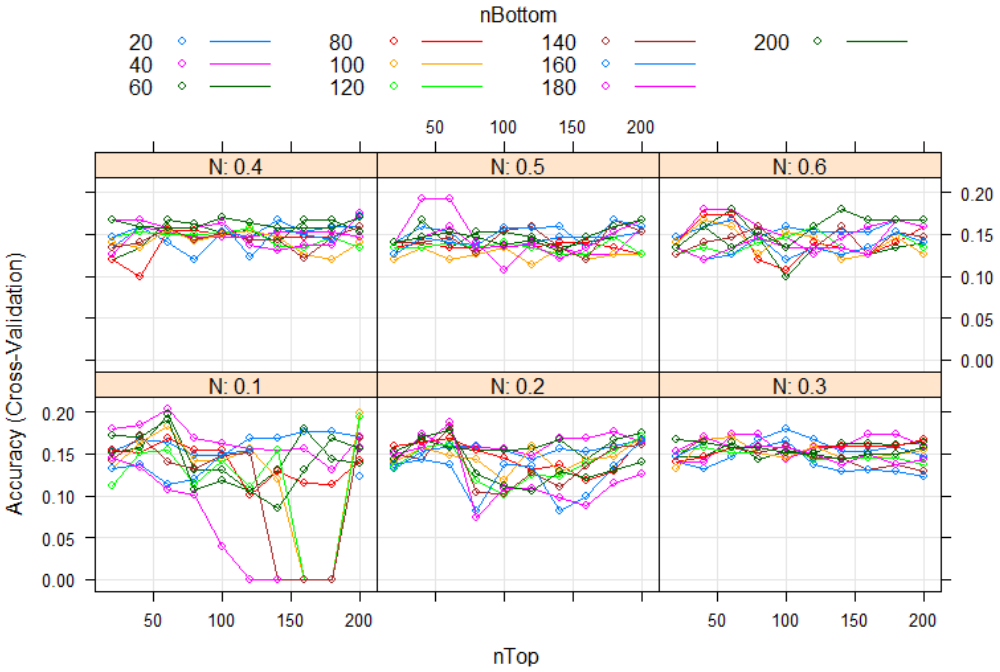


Figure 2.2: Cross-validation results of SCUDO with the grid search for parameters *nTop*, *nBottom* and *N*

**Experimental Setup** SCUDO is not designed to directly handle sequential data, therefore the best way to train the model was using the pre-computed vector of 561 features, provided by the UCI HAR dataset. These features are representative of each entire recorded sequence, and well describe their main characteristics.

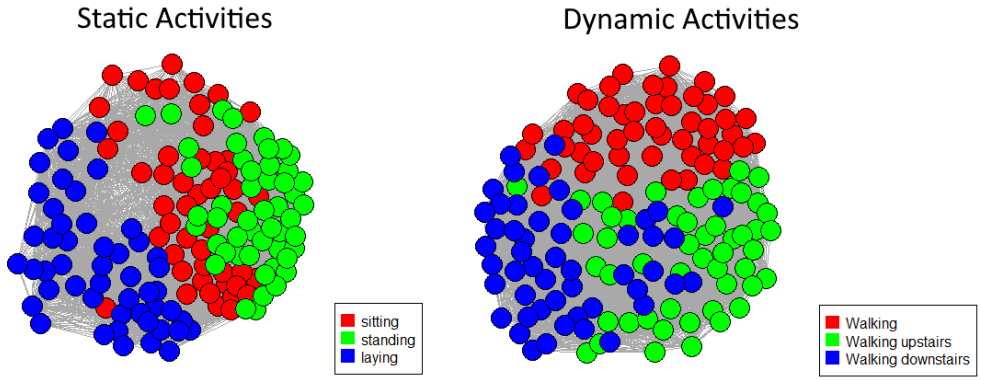


Figure 2.3: Training graphs computed by the two SCUDO models for the *static* and the *dynamic* activities

For the experiments the R version of SCUDO implemented in the package *rSCUDO* was used, and due to its quite computationally expensive training, only a small balanced subset of the data was selected. In details, 100 randomly chosen samples for each one of the six classes (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LYING) were involved to perform the training of the model, relying on the machine learning R package *caret* to perform a 5-fold cross-validation.

A preliminary feature selection was computed applying the Kruskal-Wallis test, with a p-value cut-off of 0.05. All the non-significative features were discarded from both training and testing sets.

In order to estimate the optimal tuning of the three SCUDO parameters *nTop*, *nBottom* and *N*, a grid search was performed considering all the combinations of the following parameters values:

- ***nTop***: 20, 40, 60, 80, 100, 120, 140, 160, 180, 200
- ***nBottom***: 20, 40, 60, 80, 100, 120, 140, 160, 180, 200
- ***N***: 10%, 20%, 30%, 40%, 50%, 60%

**Results** Perhaps, the cross-validation process produced poor results. As reported on the graph in Figure 2.2, even considering the best parameter configuration, $nTop = 60$, $nBottom = 180$ and $N = 0.1$, the model only obtained an average validation accuracy around the 20%.

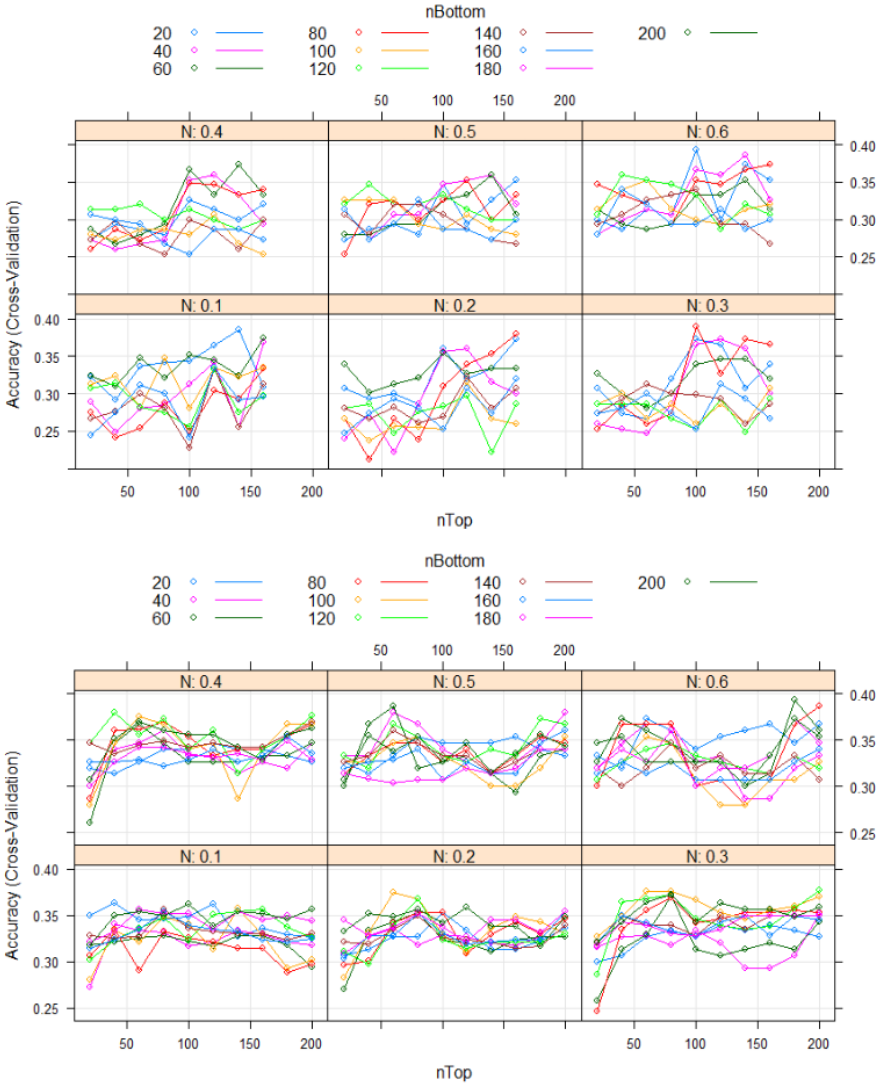


Figure 2.4: Cross-validation results produced respectively for the *dynamic* and *static* activities

The network produced by SCUDO during training is represented in Figure 2.1. From this graph we can instantly notice that the model is very effective in distinguish the two macro groups of the *dynamic* (walking, walking upstairs and walking downstairs) and *static* (sitting, standing and laying) activities, while it is not able to make a clear separation between classes belonging to the same group.

Once the training was completed an evaluation on the entire testing set was performed, in order to obtain results that could be compared with the performance of other methods. This evaluation highlighted the poor classification power of SCUDO when dealing with this type of data. The overall accuracy reached by the model was 41.7%. Despite this score is higher with respect to the one obtained during the cross-validation, it is still reflective of an insufficient predictive ability of the model.

**Macro Groups Separated Analysis** With the aim of improving the stratification of the intra-group activities, another experiment was carried out, training two separates SCUDO models, one specific for the *dynamic* and the other for the *static* activities. The setup of the experiment was identical to the previous one, selecting 100 random samples for each activity and performing a 5-fold cross-validation, applying a grid search for parameters tuning.

Even though the cross-validation accuracy increased with respect to the previous experiment (with a value around the 40% both for the *dynamic* and *static* activities, as shown in Figure 2.4), the performance of SCUDO was not satisfactory. As we can see in the two graphs produced by the models in Figure 2.3, the separation between classes belonging to the same macro group doesn't appear very clear, and this leads to samples misclassification.

### 2.1.2 Comparison with SVM

In the original paper, introducing the UCI HAR dataset [14], the authors proposed a Multi-class Support Vector Machine solution to perform the activity recognition task.

The proposed SVM make use of a Laplacian kernel instead of the most common Gaussian Radial Basis Function (RBF) version, due to hardware specific requirements. The training and the evaluation were performed using the data in the form of the precomputed 561-features vector, as in our experiments with SCUDO. The published results are reported in Table 2.1, comprehending the resulting confusion matrix, the precision and recall activity-specific scores, and the total computed accuracy obtained in the testing set.

Gaussian SVM

| Activity | Walking | Walking up | Walking down | Sitting | Standing | Laying | Recall (%) |
|---|---|---|---|---|---|---|---|
| Walking | 109 | 0 | 5 | 0 | 0 | 0 | 95.6 |
| Walking up | 1 | 95 | 40 | 0 | 0 | 0 | 69.8 |
| Walking down | 15 | 9 | 119 | 0 | 0 | 0 | 83.2 |
| Sitting | 0 | 5 | 0 | 132 | 5 | 0 | 93.0 |
| Standing | 0 | 0 | 0 | 4 | 108 | 0 | 96.4 |
| Laying | 0 | 0 | 0 | 0 | 0 | 142 | 100 |
| Precision (%) | 87.2 | 87.2 | 72.6 | 97.1 | 95.6 | 100 | 89.3 |

Laplacian SVM

| Activity | Walking | Walking up | Walking down | Sitting | Standing | Laying | Recall (%) |
|---|---|---|---|---|---|---|---|
| Walking | 109 | 2 | 3 | 0 | 0 | 0 | 95.6 |
| Walking up | 1 | 98 | 37 | 0 | 0 | 0 | 72.1 |
| Walking down | 15 | 14 | 114 | 0 | 0 | 0 | 79.7 |
| Sitting | 0 | 5 | 0 | 131 | 6 | 0 | 92.2 |
| Standing | 0 | 1 | 0 | 3 | 108 | 0 | 96.4 |
| Laying | 0 | 0 | 0 | 0 | 0 | 142 | 100 |
| Precision (%) | 87.2 | 81.7 | 74.0 | 97.8 | 94.7 | 100 | 89.0 |

Table 2.1: SVM classification results

In our replication of the experiment, we decided to opt for the RBF kernel implementation, since it provided a slightly improvement in the performance, and we were not affected by any particular hardware limitation.

The experiment was carried out in the Python environment, relying on the package *scikit-learn* to implement and evaluate the SVM model, preforming a 5-fold cross-validation (Figure 2.5). The obtained results were reflecting the ones reported in the original work, with an accuracy on the testing set of 89.5%. The training of the SVM was very computationally efficient, especially when compared to SCUDO, resulting in an overall better performance.

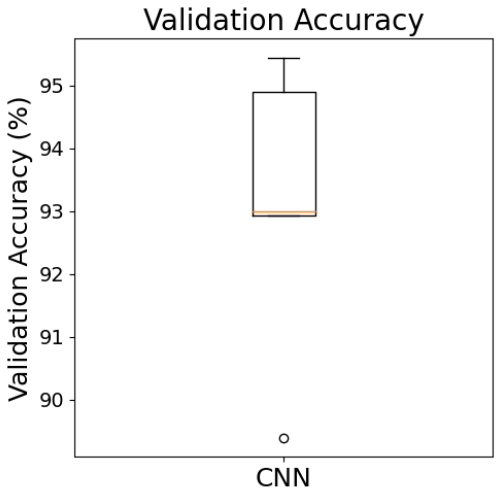


Figure 2.5: Validation accuracy obtained in the 5-fold cross-validation.

### 2.1.3 HMM Ensemble Model

The main issue in classifying time sequences data with the more traditional machine learning algorithms is represented by their inability to capture the time-dependent information that is implicitly contained in the data. As we have experienced both in the SCUDO and SVM experiments, the computation of features that can describe the characteristics of each entire sequence represented an essential procedure, in order to successfully fit the models.

One of the possible solutions for this problem is implementing a Hidden Markov Model-based method, in order to encapsulate the sequential structure of the data. In particular, we focused on the HMM-Ensemble model proposed in [7]. As suggested by the name, this model is composed by and ensemble of HMMs separately trained for each class. The probabilities inferred by the set of models are then involved into a decision process, in order to make final predictions.

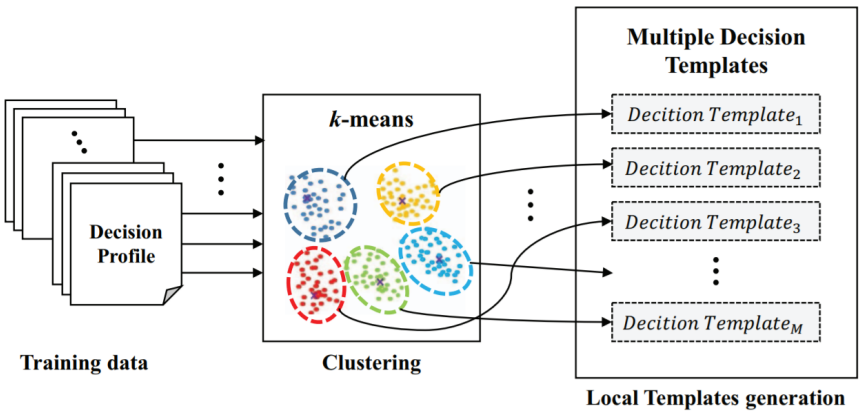


Figure 2.6: Creation of DTs using the K-Means algorithm.

**HMME architecture** The pipeline of the proposed method is well-described in Figure 2.7. The triaxial acceleration signals are taken in input. An initial subsampling of the original data is performed, in order to decrease the length of the observation sequences: each sequence, originally composed by 128 reads, is split in 8 subsequences of length 16 reads, and for each of them mean and standard deviation are computed.

Every input signal is therefore transformed into a new sequence of 8 mean and standard deviation couple:

$$O^k = \{(\mu_1^k, \sigma_1^k), ..., (\mu_8^k, \sigma_8^k)\}$$

where $k$ represents the acceleration axis (x, y or z).

Those resulting sequences are subsequently discretized applying the K-Means (k = 64) clustering algorithm on the entire training set of mean and standard deviations couples. The resulting sequences are in the form:

$$O^k = \{a_1^k, ..., a_i^k, ..., a_8^k\}$$

with $a_i^k = 0, 1, 2, .., 63$, indicating the assigned cluster for the couple $(\mu_i^k, \sigma_i^k)$.

All the training sequences are grouped by class and axis, and for each resulting group a Multinomial HMM model, with 10 hidden states, is trained through the Baum-Welch algorithm. The resulting ensemble of HMMs is then applied to compute the a posteriori probability estimate for each input sequence, relying on the Viterbi algorithm. All the probabilities computed for each input record can be represented in a matrix called Decision Profile (DP):

$$DP(O) = \begin{pmatrix} P(O^x|\lambda_1^x) & ... & P(O^x|\lambda_i^x) & ... & P(O^x|\lambda_c^x) \\ P(O^y|\lambda_1^y) & ... & P(O^y|\lambda_i^y) & ... & P(O^y|\lambda_c^y) \\ P(O^z|\lambda_1^z) & ... & P(O^z|\lambda_i^z) & ... & P(O^z|\lambda_c^z) \end{pmatrix}$$

where each element $P(O^k|\lambda_i^k)$ represents the estimated probability of observing the sequence $O^k$, given the HMM model with the set of parameters $\lambda_i^k$. Those parameters are the ones estimated during training with the Baum-Welch algorithm, using sequences of class $i$, in $\{1, 2, .., c\}$, and acceleration in the axis $k$, in $\{x, y, z\}$.

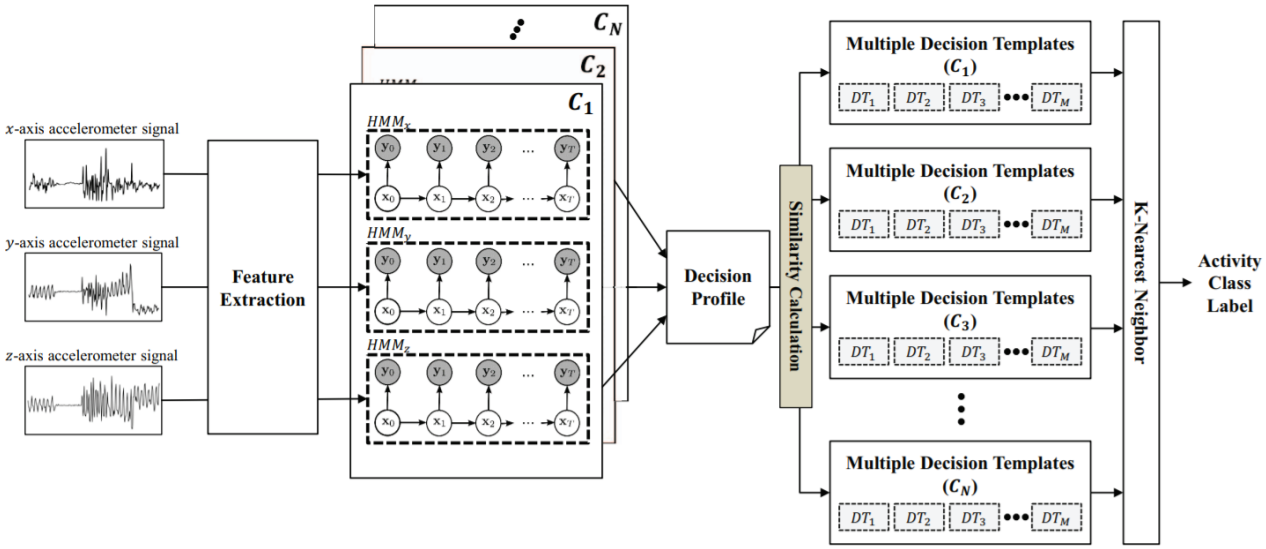


Figure 2.7: Architecture of the HMME model.

Once the DPs are computed for all the training sequences, the clustering algorithm K-Means is applied once again, grouping all the DPs belonging to the same class into 64 clusters, as shown in Figure 2.6. The centroids of those clusters are saved in the so-called Decision Templates (DTs). Each DT can be seen as a point in the new feature space created by the ensemble of HMMs. As explained in [7], the choice of computing multiple DTs for each class has the aim of capturing intraclass variations between the DPs belonging to the same group.

After the training is completed, the class of the testing sequences are predicted, computing the correspondent DPs through the HMM Ensemble, and subsequently applying the K-Nearest-Neighbours (K-NN) algorithm to perform the final classification.

K-NN with $k = 3$, finds the 3 *DTs* that are closer to the target *DP*, and the most represented class is finally assigned.

**Replication details** The authors didn't provide any source code, therefore a new implementation was needed in order to obtain a replication. The experiment was carried out in a Python environment, relying on the *hmmlearn* library to implement the HMM models and on the package *scikit-learn* for the other Machine Learning techniques (K-Means and K-NN) and utilities.

The same training and evaluation procedures described in the original paper were followed, in order to have a fair comparison of the results.

As described in [7], the experiment consists of 10 repetitions, in which the data are randomly split in 70% training and 30% testing set. All the parameters were set as follow:

- $K = 64$ for the K-Means algorithm used in the discretization of the features.
- $K = 64$ for the K-means algorithm applied to group the DPs into the corresponding DTs.
- $n\_states = 10$, the number of hidden states for each HMM model
- $K = 3$ for the K-NN algorithm, to perform the final classification

**Results** Despite the fact our replication followed in the details the architecture described in the original paper, the obtained results didn't reflect the ones published. In fact, our average validation accuracy over the 10 experiments was around 76.5% (Figure 2.8), versus the 83.5% published by the authors.

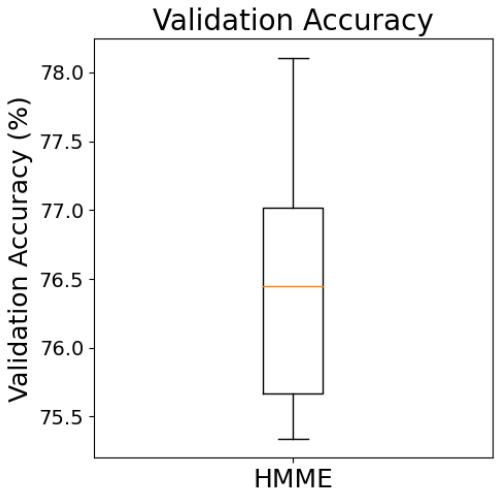


Figure 2.8: Validation accuracies of the 10 repetitions.

This difference in the performance could be due to some missing information regarding the implementation, or the selection of a specific subset of data to preform training and evaluation. Without having access to the original source code it was not possible to fully understand the cause of the discrepancy.

The model that best performed on the validation set during the 10 experiments was then selected and evaluated on the predisposed testing set. The results of the evaluation are reported in the following table:

| Activity | Precision | Recall |
|---|---|---|
| Walking | 0.70 | 0.55 |
| Walking upstairs | 0.55 | 0.84 |
| Walking downstairs | 0.62 | 0.54 |
| Sitting | 0.67 | 0.74 |
| Standing | 0.79 | 0.74 |
| Laying | 0.99 | 0.80 |
| **Total Accuracy** | | 0.71 |

The accuracy obtained on the testing was slightly lower than the one obtained during validation, with a score of 71%.

**Improved HMME Version** With the purpose of improving the performance of the HMME model some modifications were applied to the original architecture. In particular, we modified some of the parts, that in our opinion represented critical points, affecting the discriminative power of the algorithm. The new scheme of the pipeline is shown in Figure 2.9

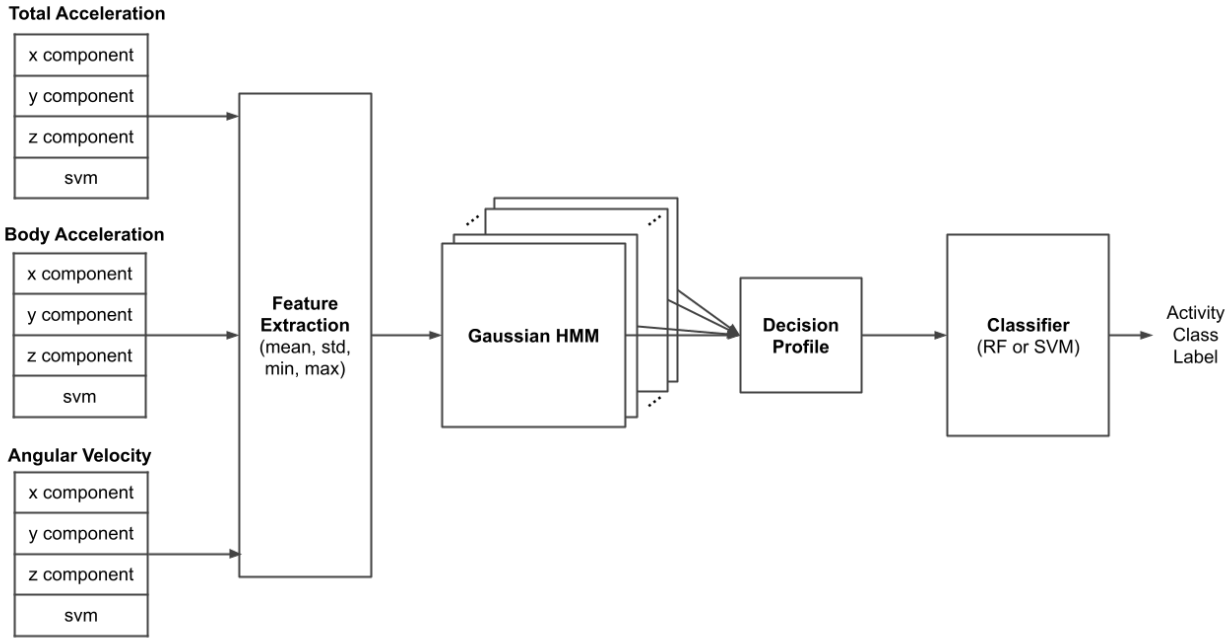


Figure 2.9: New architecture of the improved HMME model.

The UCI HAR Dataset contains three different types of signals: the *total* 3-axial acceleration (comprehending the gravitational component), the *body* 3-axial acceleration (in which the gravitational component is removed) and the 3-axial angular velocity. While the original architecture was designed to work only with one 3-axial acceleration signal, we modified the model in order to integrate all the three types of sequences. Furthermore, for each of them the signal vector magnitude (svm) was computed as:

$$svm = \sqrt{x^2 + y^2 + z^2}$$

This implied quadruplicating the number of HMM models to be trained, incrementing the overall computational cost, but also adding a supplemental amount of information that could be potentially crucial in improving the discriminative power of the model.

The next modification regarded the features discretization process, originally executed by the K-Means algorithm. As previously described, in this procedure the mean and standard deviation couples are grouped by similarity, in 64 different clusters. The resulting labels are used to create new sequences of discrete values, subsequently employed in the Multinomial HMM training. Our main criticism about this procedure is the further loss of information caused by the discretization of the features. Furthermore, the choice of setting K-Means with $K = 64$ was not justified by the authors. To bypass this procedure, we tried to substitute the Multinomial HMM with its Gaussian version, in which the emission probabilities are generated from a Normal distribution, with mean and covariance parameters estimated during the Baum-Welch training procedure. This allowed us to fit our HMM models giving in input directly the sequences of computed features composed by continuous values, instead of discretizing them.

In addition to mean and standard deviation, also the minimum and maximum values were computed for each subsequence, with the aim of obtaining additional information regarding each subsequence. Other types of variables could have been further added, however we decided to keep the feature space low dimensional, with the aim of not further overloading the model training procedure.

The last changing was applied to the design of the final decision process. Instead of computing a set of decision templates and make predictions using K-NN, we opted to train more complex models, such as Random Forest and SVM, on the entire set of computed DPs. This allowed us to avoid the construction of the DTs, representing an additional level of data abstraction, and therefore bypassing the use of the K-Means algorithm, in which the choice of the parameter value $K = 64$ was once again not justified in the paper.

**Results of the improved version** While some of the modifications applied to the model didn't affect its performance, others were particularly effective, and significantly improved its discriminative power. In particular, the involvement of the Gaussian HMM model resulted in slightly decrease of the performance, while the addition of complex model such as Random Forest and SVM for the final prediction led to an increase of the classification results.

After testing the effect of each applied changing, the architecture that resulted to be the best performing was composed as follows:

- **Input signals:** *total* 3-axial acceleration, *body* 3-axial acceleration and 3-axial angular velocity, plus the three corresponding signal vector magnitudes.
- **Computed features:** mean, standard deviation, maximum and minimum values. The features were standardized as in the original model using the k-Means algorithm with $K = 64$.
- **HMM model:** Multinomial HMM.
- **Final decision process:** classification with a Random Forest model, trained on the entire training set of computed DPs.

This new model was evaluated in the same manner of the original method. The average accuracy obtained in the validation sets of the 10 experiments was 96.2% as shown in Figure 2.10.

The performance scores obtained on the testing set are in the following table:

| **Activity** | **Precision** | **Recall** |
|---|---|---|
| Walking | 0.92 | 0.92 |
| Walking upstairs | 0.89 | 0.96 |
| Walking downstairs | 0.94 | 0.87 |
| Sitting | 0.82 | 0.83 |
| Standing | 0.85 | 0.84 |
| Laying | 0.99 | 0.98 |
| **Total Accuracy** | | 0.90 |

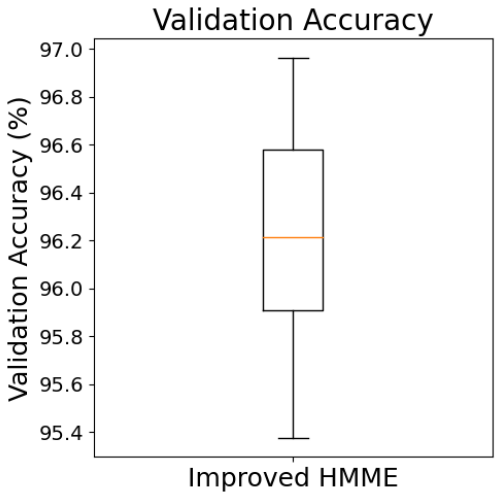


Figure 2.10: Validation accuracies of the 10 repetitions.

The new architecture of the HMME model outperformed, not only our replication of the original work, but also the published results, obtaining satisfactory performances during both validation (avg. accuracy 96.2%) and testing (accuracy 90.0%) processes.

### 2.1.4 Deep Learning Approach

One of the most promising fields in Machine Learning for the analysis of complex structured data, such as images and sequences, is the represented by the Artificial Neural Networks (ANN). The popularity of these models has exploded in the recent years, becoming the state of the art for modelling such type of data.

Different Neural Networks have been already applied on the problem of Human Activity Recognition. In [13] a comparison between multiple architectures was made, evaluating their classification performance on a dataset similar to the UCI HAR.. The evaluated models are:

- **Long Short-Term Memory (LSTM)**, the most common type of Recurrent Neural Network (RNN), specifically designed to model sequential data.
- **Bi-directional LSTM (BiLSTM)**, a type of LSTM in which sequences are processed in both directions (forward and backward).
- **1-Dimensional CNN (1D-CNN)**, a Convolutional Neural Network that applies convolution on 1-Dimensional data (see chapter 1.2.5).
- **Convolutional LSTM (ConvLSTM)**, an hybrid model combining a CNN and an LSTM architecture.

Comparing the results reported in the paper the CNN architecture resulted the best performing model. Therefore, we decided to test this type of network on the UCI HAR dataset in order to make a comparison with the previously evaluated algorithms.

**Architecture and implementation details** In the first place a shallow architecture similar to the one proposed in [13] was created. Its structure is shown in Figure 2.11.

The first block is composed of a 1D-Convolutional Layer with 64 filters, kernel size 5 and *ReLU* activation function, a Max Pooling layer with pool size 2 and a Dropout layer with rate 0.5.

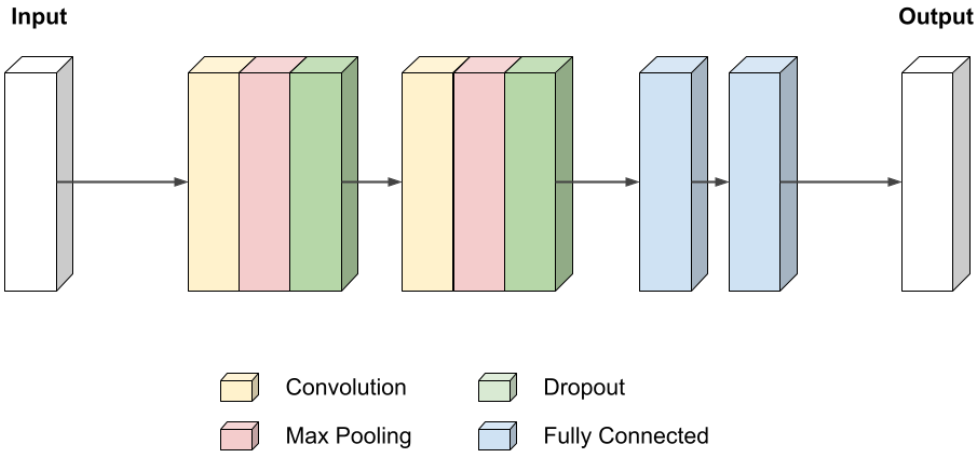


Figure 2.11: Architecture of the applied CNN model.

The second block has the same structure of the first one, while the final two layers are fully connected: one with 128 output units and *ReLU* activation function and the other with 6 output units (as the number of classes to predict) and *softmax* activation.

The chosen loss function to minimize was the *Categorical Cross-Entropy Loss*, computed as:

$$Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i$$

where $\hat{y}_i$ is the i-th output predicted by the model and $y_i$ is the corresponding ground truth value, while $n$ represents the number of outputs.

The implementation was done in Python, using the *TensorFlow* and *Keras* libraries to create and train the network. The execution was run in the Google Colab environment, taking advantage of a dedicated GPU, to considerably speed up the training procedure.

The optimization of the parameters was performed using Adam [8], with an initial learning rate equal to 0.001. This type of optimizer represents an enhancement of the traditional Stochastic Gradient Descent, in which the learning rate is no more constant, but it is instead adapted over time, proportionally to the gradient magnitude.

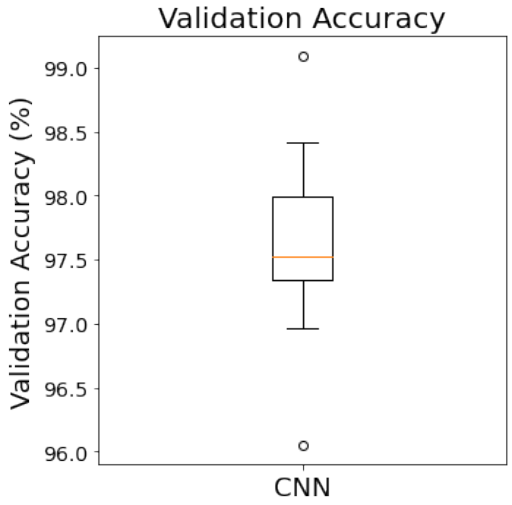


Figure 2.12: Validation accuracies of the 10 repetitions.

**Results** The experiment was carried out following the same pipeline used for evaluating the HMME model. 10 repetitions have been executed, each time randomly splitting the data in 70% training and 30% validation sets. The best parameter setting was then selected to be evaluated on the testing set.

Before training the network, the data $X$ was standardized in the interval [-1,1], adjusting by mean $\mu$ and dividing by standard deviation $\sigma$:

$$\hat{X} = \frac{X - \mu}{\sigma}$$

Each model was trained for 50 epochs, applying an early stoppage in the case of any substantial improvement hasn't been experienced after 10 subsequent epochs. The values of the loss function and the classification accuracy obtained in the 10 experiments, during training and validation, are summarize in Figure 2.13. The average validation accuracy was of 97.5% as shown in the boxplot in Figure 2.12.

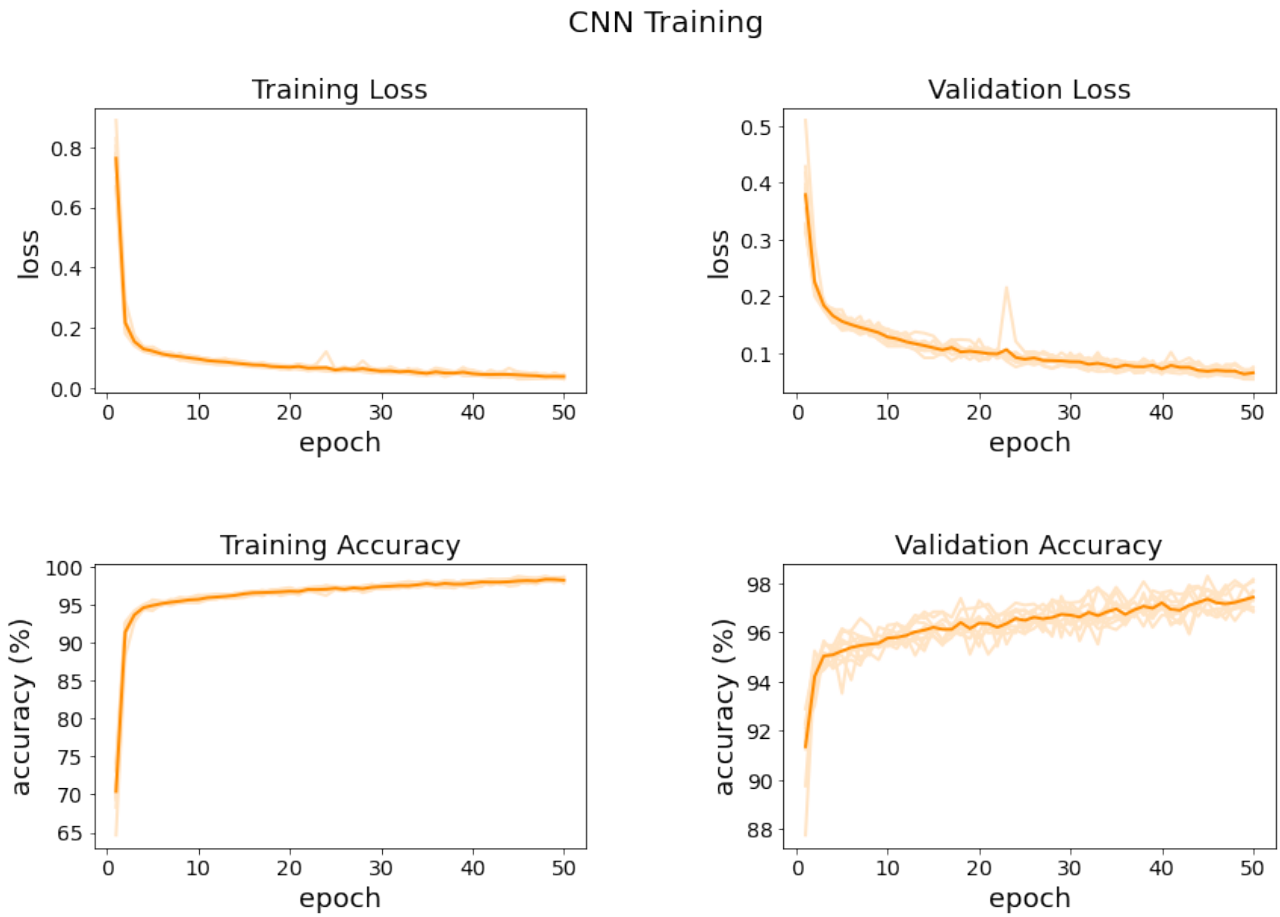


Figure 2.13: Progress of loss and accuracy values over the training epochs. The behaviour of each experiment is plotted in shaded orange, while the brighter line corresponds to the mean value computed among all the experiments

The best parameter setting to be evaluated in the testing set was selected in terms of minimization of the validation loss function. The classification accuracy reached by the model was 91.0%. As for the other methods, the computed evaluation metrics are reported in the table:

| Activity | Precision | Recall |
|---|---|---|
| Walking | 0.99 | 0.96 |
| Walking uostairs | 0.99 | 0.94 |
| Walking downstairs | 0.74 | 0.82 |
| Sitting | 0.74 | 0.82 |
| Standing | 0.83 | 0.73 |
| Laying | 1.00 | 1.00 |
| **Total Accuracy** | | 0.91 |

**1D-ResNet** With the aim of further improving the classification performance, another deeper CNN architecture, called ResNet [6], was trained on the UCI HAR data. This CNN is widely used in the field of image classification, but its 1-dimensional version has been successfully applied to model sequential data [22].

In general, deep networks have the tendency of building too complex models, causing the so-called Degradation problem, in which accuracy gets saturated, often resulting in an increased training error. Another common problem that affects deep architectures is the gradients vanishing, in which gradients progressively shrink to zero after few optimization steps, causing the weights to be never updated. The idea behind ResNet is trying to solve these issues designing a so-called residual connection, that allows to bypass entire blocks of the network, and acting as a gradient booster.

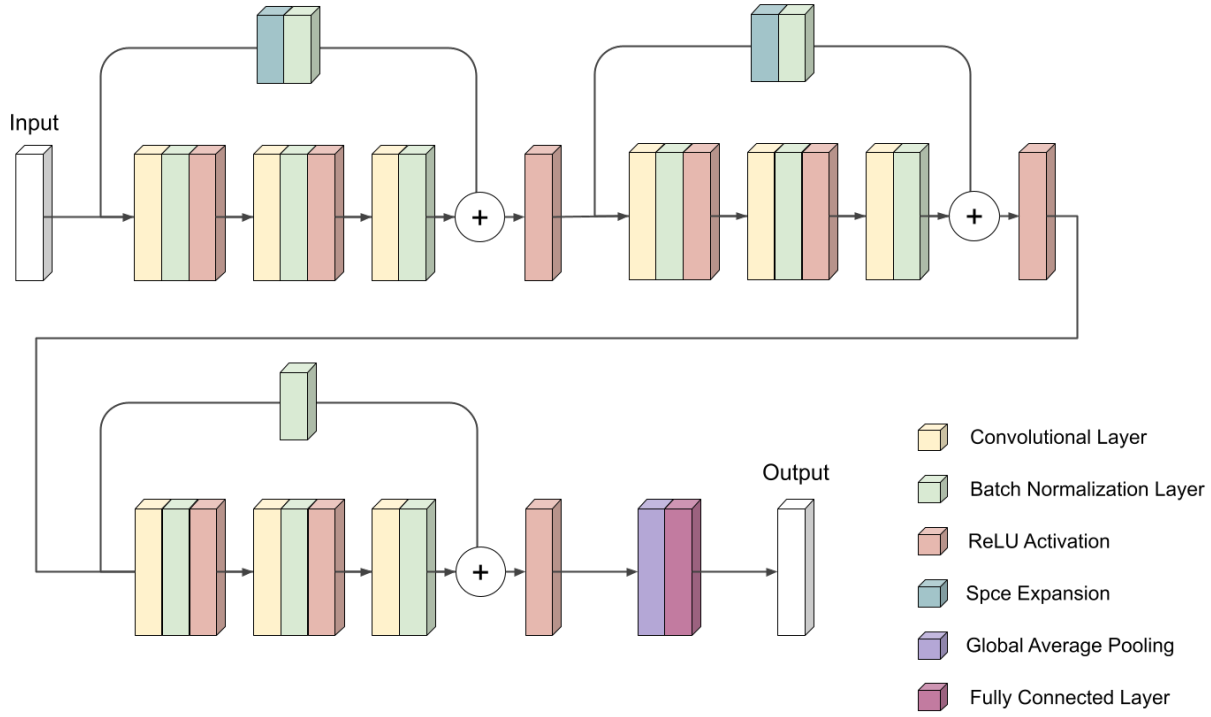


Figure 2.14: Architecture of the applied ResNet model.

The implemented architecture of the 1-D ResNet was similar to the one proposed in [22], and in shown in Figure 2.14. It is composed of three convolutional blocks, followed by a final dense layer to extract the output.

In details, the structure of the four blocks is:

- **Block 1**
  - Convoultional layer with 64 filters, kernel of size 8.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 64 filters, kernel of size 5.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 64 filters, kernel of size 3.
  - Batch Normalization layer.
- **Block 2**
  - Convoultional layer with 128 filters, kernel of size 8.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 128 filters, kernel of size 5.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 128 filters, kernel of size 3.
  - Batch Normalization layer.
- **Block 3**
  - Convoultional layer with 128 filters, kernel of size 8.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 128 filters, kernel of size 5.
  - Batch Normalization layer followed by ReLU activation function.
  - Convoultional layer with 128 filters, kernel of size 3.
  - Batch Normalization layer.
- **Final block**
  - Global Average Pooling
  - Fully Connected layer with 6 output units (as the number of classes to predict).

Before every Convolutional layer the input is zero padded in order to maintain its shape after the convolution. Each residual (or identity) connection is made by applying a Convolutional layer, with kernel size equal to 1, only in the case the input has a different shape with respect to the relative block's output (connections 1 and 2). Next, batch normalization is computed, and the resulting vector is added to the output of the corresponding block, and finally applying a ReLU activation function.

**Results** The same experimental procedure carried out for the CNN was followed for the 1D-ResNet. The progress of the training and validation performances over epochs are reported in Figure 2.15. As it's possible to notice, ResNet reaches the plateau on the validation test in a faster way with respect to the shallow CNN. This means that the model starts overfitting the training data after few steps, therefore a smaller number of training epochs may be sufficient. The obtained validation accuracies are shown in Figure 2.16.

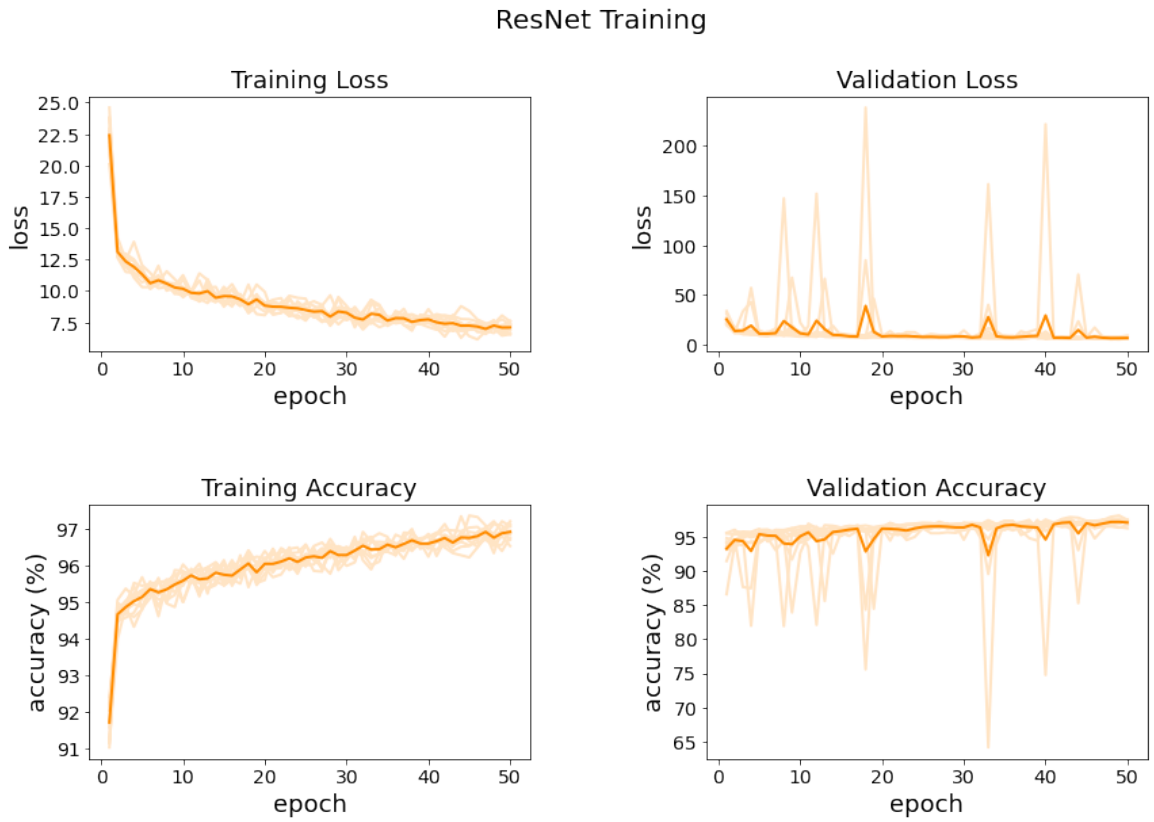


Figure 2.15: Progress of loss and accuracy values over the training epochs.

As well as the shallow CNN architecture, the best performing parameters setting obtained though the epochs was selected and the model was evaluated on the testing set. The results are reported in the following table.

| **Activity** | **Precision** | **Recall** |
|---|---|---|
| Walking | 1.00 | 1.00 |
| Walking uostairs | 0.99 | 0.94 |
| Walking downstairs | 0.99 | 0.99 |
| Sitting | 0.84 | 0.84 |
| Standing | 0.87 | 0.89 |
| Laying | 0.99 | 1.00 |
| **Total Accuracy** | | 0.94 |

As reflected by the results, even though the performance of the two tested models were very similar, in terms of average validation accuracy (97.5% for the shallower CNN and 97.3% for the ResNet), the ResNet architecture better performed on the testing set data, with an accuracy score of 94%.

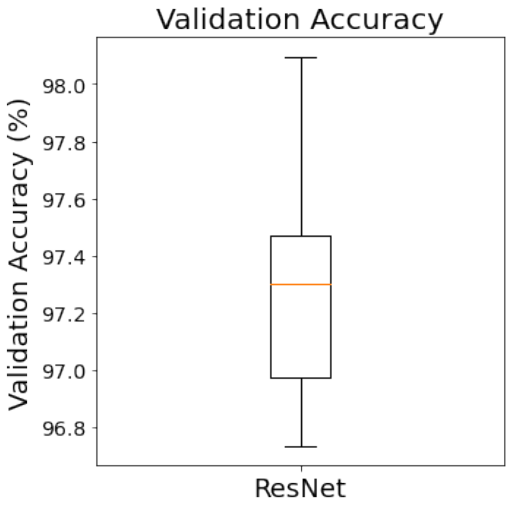


Figure 2.16: Validation accuracies of the 10 repetitions.

## 2.2 Parkinson's Disease Detection

The task of Parkinson's Disease (PD) detection consists in performing a preliminary diagnosis of the subjects condition, analysing the acceleration and angular velocity data recorded by sensors worn during a walking test. The mPower application provided three different types of sequences, called *outbound*, *return* and *rest*, representing the three stages of each walking experiment.

As a starting point for our analysis, we decided to replicate the work produced by the *Dream Challenge*'s winning group, described in [25]. The choice of this particular model was not only driven by the fact it obtained the best results in the challenge. Indeed, the proposed model is a CNN-based method, with a baseline similar to the ones we had already experienced during our Human Activity Recognition experiments.

The solution proposed by the authors is an ensemble of CNN models, with a Random Forest as final classifier, following the same rational we encountered in the HMME model.

### 2.2.1 Model Architecture

The CNN architecture used for feature extraction is composed of 8 convolutional layers with number of filters respectively equal to 8, 16, 32, 32, 64, 64, 128, 128 and kernel size equal to 5, 5, 4, 4, 4, 4, 4, 5. Each convolutional layer is followed by a ReLU activation function and a Max Pooling layer with pool size 2. A final fully connected layer is subsequently connected to the flatten result of the entire convolutional process, applying a sigmoidal activation to produce the final binary classification. The chosen loss function to minimize is the binary Cross-Entropy loss, computed as:

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log(\hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

where $\hat{y}_i$ is the i-th output predicted by the model and $y_i$ is the corresponding ground truth value, while $n$ represents the number of samples.

For each type of sequence, *outbound*, *return* and *rest*, 5 CNN models were trained on different sets of data, randomly splitting each time the in 50% training and 50% validation sets. The obtained CNN ensemble is composed of 15 different models. These are subsequently used to perform prediction on all the training samples. The resulting vectors of 15 predicted values is then given in input to train the Random Forest Classifier.

### 2.2.2 Data Transformation

Before training, the data are pre-processed applying different techniques. First, a down sampling of the sequences is applied, transforming the original frequency of 100Hz to 50Hz. Each signal is then standardized in the range $[-1, 1]$, and since sequences may have different sizes, they are zero-padded or cut in order to obtain a common length of 4000 reads.

One crucial point highlighted in the paper is the process of data augmentation, applied to the training set. This procedure permits to decrease the effect of overfitting, randomly transforming the data with different methods. In details, the two following augmentation techniques were applied:

- **Random Scaling**. Each sequence is randomly time-scaled by a factor between 0.8 and 1.2. This augmentation technique simulates chaining in the sampling rate.
- **Random Rotation**. Each 3-axial signal is randomly rotated with the aim of simulating different smartphone positions. The new vector is computed generating a random rotation matrix and multiplying it to the original vector.

### 2.2.3 Replication Details

The replication code was provided by the authors and is publicly available in a GitHub repository. It was implemented combining Perl scripts, for data handling, and Python code for the creation, training, and evaluation of the model. The Deep Learning libraries used to build and train the CNNs were Theano and Lasagne.

With the aim of improving the computational time of the experiment we decided to create a de novo implementation. Since the previously used libraries required a deprecated version of Python (2.7), the new implementation was created using Tensorflow and Kears, allowing also to easily transfer the model to the Google Colab environment, and allocate the resources on a GPU device.

The replication followed exactly the same evaluation procedure described by the authors. The data were split in 50% train and 50% test sets, making and individual-wise division in order to obtain a more reliable evaluation. Each CNN model was then trained on a randomly selected 50% of training data and evaluated on the remaining validation set. This time the division was simply made record-wise.

### 2.2.4 Experimental Results

Our replication of the experiment produced similar results to the ones published in the paper. The individual-wise AUC-ROC score obtained was 81.0%, versus the original 85.58% reported by the authors. This decrease in the performance can be due to the bootstrap resampling, performed in the original work to rebalance the dataset. In fact, the details about this procedure are not clearly explained in the paper, therefore it was not possible to obtain an exact replication of the selected training set.

One big booster for the algorithm performance was the so-called "pulling procedure". In this process the record-wise predictions computed by the set of CNNs are grouped by individual, and only the maximal values obtained for each type of sequence were considered to perform the final classification. The authors' hypothesis justifying the increment of the model performance, when considering only the maximal values, is that the most severe records may be more representative of the disease phenotype. From our experiments we concluded that this procedure was crucial in enhancing the model performance, since the AUC-ROC score, computed by record instead of by individual, was only around 65% (the score reported in the original work exceeds the 70%).

### 2.2.5 Improvement Attempts

Despite the good results obtained in the individual-wise classification, the model was not particularly accurate in the prediction of single records. This is reflected by the poor performance of each individual CNN in the ensemble.

In order to improve the record-wise predictive power, different approaches were tested, modifying the original structure of the CNN model composing the ensemble.

In the first place, we analysed the difference in terms of validation AUC between the originally proposed CNN model and the 1-D ResNet, already evaluated in the HAR task.

The architecture of the 1-D ResNet was the same described in chapter 2.1.4, substituting only the final softmax activation function with a sigmoidal activation, aiming to perform binary classification. The dataset was randomly divided in 70% training and 30% validation sets and the two models were trained for 50 epochs on all the three types of records (outbound, rest and return). The results obtained in our test is shown in the following table:

| | **AUC-ROC** | | | **Avg. Computational Time (seconds per epoch)** |
|---|---|---|---|---|
| | Outbound | Rest | Return | |
| **Original CNN** | 68.6 | 71.6 | 69.9 | 7 |
| **1-D ResNet** | 79.3 | 75.2 | 74.9 | 57 |

As we can see, the ResNet architecture provided better performances in the validation process, but with the drawback of drastically increasing the computational time. For this reason, it became impractical to perform multiple repetitions of the experiment, or apply a cross-validation procedure, to better evaluate the models.

Other strategies were subsequently involved, trying to further improve the model's predictions.

**1-D RestNet + HMM classifier** The first attempt consisted in integrating the features extracted by the ResNet with an HMM-based classifier, similar to the HMME discussed in chapter 2.1.3.

The idea behind this method is training the HMM models on the set of features extracted from the network, since the sequential structure of the data should be somehow maintained during the convolutional process. The complete architecture of the new model is shown in Figure 2.17 and is similar to the one proposed in [21].

In details, the dataset was randomly split by individual in 50% training and 50% test sets, and from the training data a 30% was selected as validation. Three ResNet models were then trained for 30 epochs and evaluated on the validation data. The configurations of the three models that best performed during the 30 epochs were subsequently selected and utilized to extract the features from the validation records. The features were obtained discarding the last dense layer of the network, directly obtaining the output of the final Global Average Pooling. For each one of the three groups of extracted features two Gaussian Mixture Model-HMM (GMM-HMM) were fitted, selecting respectively only the "control" and the "affected" records. In this type of HMM the emission probabilities are estimated from a Gaussian Mixture Model (GMM)[23]. Each HMM was composed of 8 hidden states, while the GMMs were built with 3 components.

During inference time on the testing set, the corresponding features were once again extracted by the three ResNets, and the set of GMM-HMM models computed the a posteriori probability estimates for each testing record.

All the obtained probabilities were divided in the two classes and the total sum was computed as:

$$P^i_{total} = \sum_{\forall s \in S} P(O_s|\lambda_{s,i})$$

where $i \in \{"control","affected"\}$ represents the class label and $s \in \{"outbound","rest","return"\}$ the type of record. $P(O_s|\lambda_{s,i})$ is the probability estimate for the observation $O_s$ given the set of model parameters $\lambda_{s,i}$, specifically estimated during training for records of type $s$ in the class $i$

Finally, the class prediction was made, accordingly to the maximum between the total "control" and "affected" probabilities previously computed:

$$Pred = \underset{i}{\operatorname{argmax}}\, P^i_{total}$$

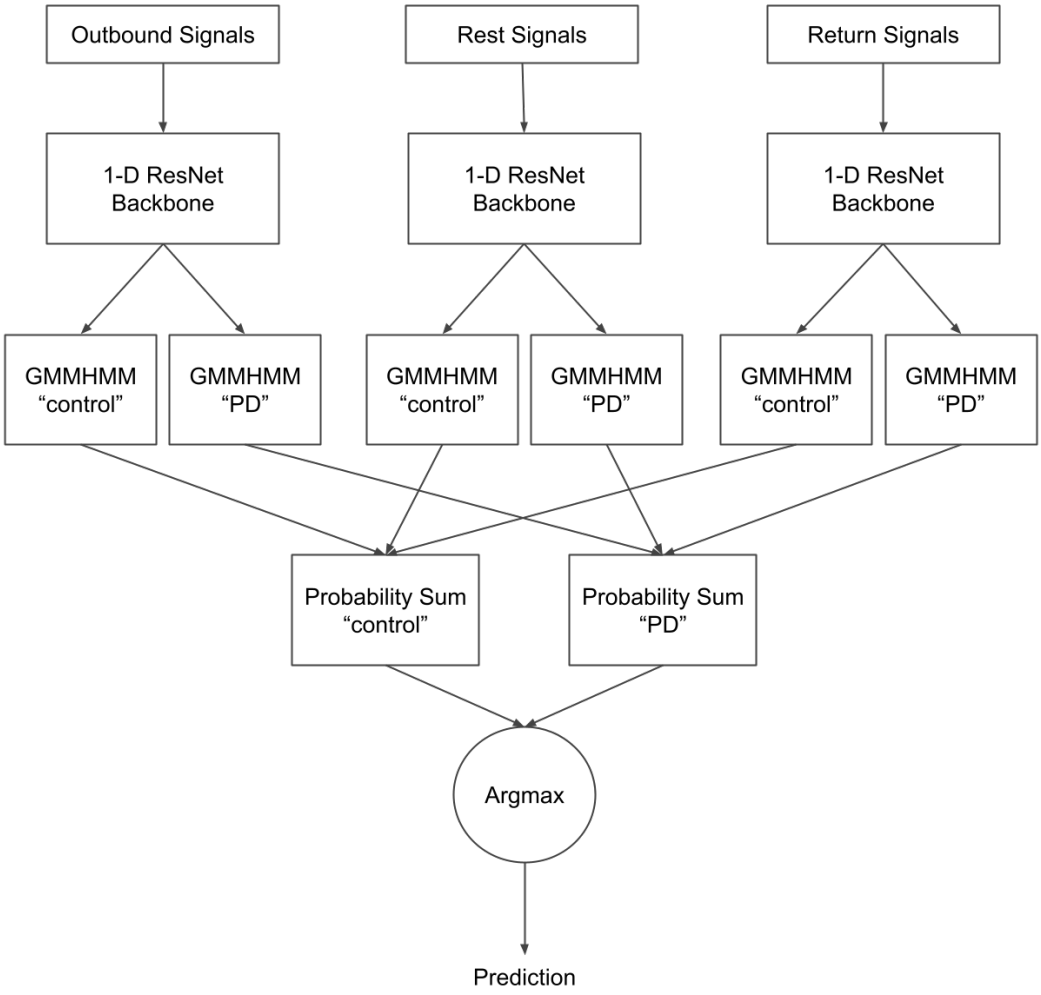


Figure 2.17: 1D-ResNet + HMM classifier architecture.

This process represents a simplification of the HMME decision procedure, since in the original model a specific classifier needed to be trained on the vectors of computed probabilities, enhancing its discriminative power, but at the same time incriminating the overall computational cost.

**1-D RestNet + RF and SVM** The second type of tested model consisted in substituting the last dense layer of the trained ResNet with another type of classifier, such as a Random Forest or a Support Vector Machine. The structure of the model is described in Figure 2.18.

In details, the division of the dataset followed the same procedure as the previous experiment. Three different ResNet models were trained for 30 epochs, respectively on the outbound, rest and return training sequences. Subsequently, the sets of features were extracted from the validation data, and they were concatenated into a single vector, in order to fit the RF/SVM model.

During testing, the data features were extracted, and the final prediction was simply made by the trained classifier.

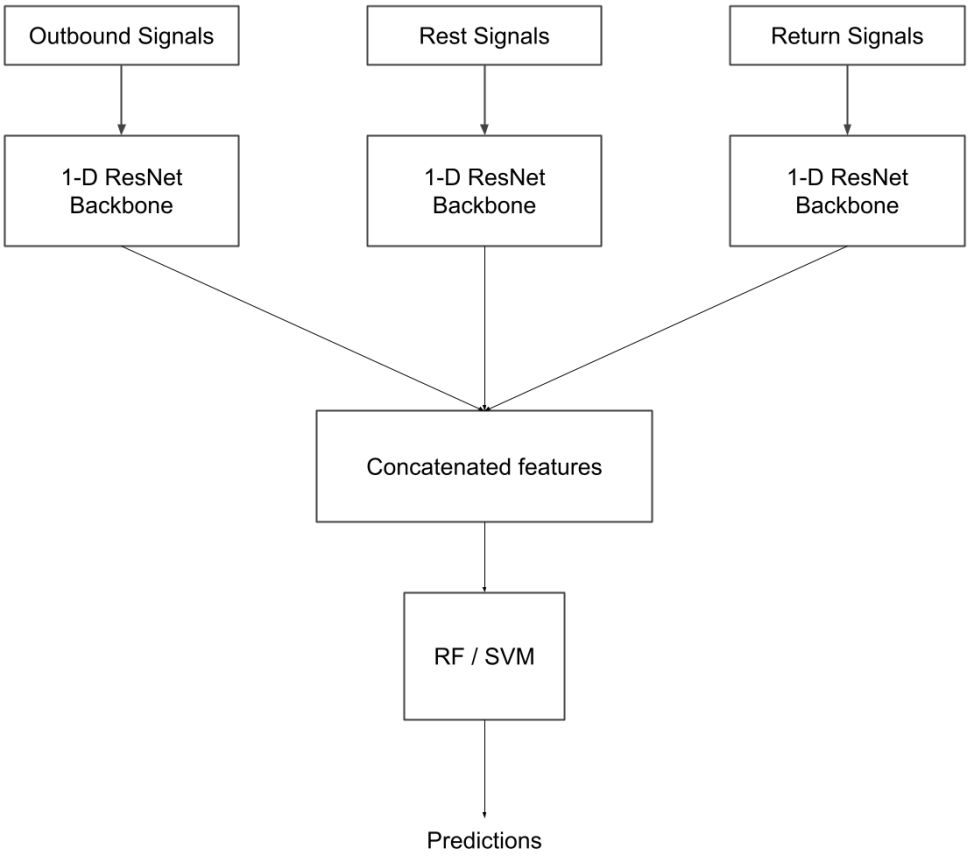


Figure 2.18: 1D-ResNet + RF/SVM classifier architecture.

**Models Evaluation** The previously described methods were evaluated on their predictive performance on the testing set. In the following table are reported the metrics computed for each model.

| | **Precision** | **Recall** | **Accuracy** | **AUC-ROC** |
|---|---|---|---|---|
| 1-D ResNet | 63.8 | 68.3 | 59.4 | 57.7 |
| 1-D ResNet + HMM | 40.5 | 7.0 | 40.4 | 46.4 |
| 1-D ResNet + RF | **69.3** | 68.3 | 64.3 | **63.6** |
| 1-D ResNet + SVM | 68.4 | **72.4** | **64.8** | 63.4 |

As shown, the HMM-based solution didn't provide any improvement, leading instead to a decrease in the performance. On the other hand, the integration of both SVM and RF in the neural network revealed to be a good strategy, and significantly increased the predictive power of the single ResNet.

**Data Selection Experiment** One of the main issues encountered in the Parkinson's Disease Detection task is in our opinion represented by the structure of the dataset. mPower indeed contains different imbalances, in particular in terms of class, age, gender and number of recorded samples per individual. In details, individuals with age $< 45$ are not represented in the PD group and over-represented in control group. Moreover, the number of samples recorded for each individual is very variable and this can affect both training and evaluation processes, especially when evaluation is performed individual-wise.

Considering those issues, a subset of data was created, selecting individuals with age $> 45$ and number of samples greater than 2. From the resulting set only 5 samples per individual were selected, randomly oversampling records in the case less than 5 recorded sequences were present.

The originally proposed model was so tested on the new filtered subset, with the aim of preventing it to lean bias, and therefore determine how the dataset imbalance, especially in terms of number of records per individual, affected its performance.

To evaluate the model the same originally proposed pipeline was followed. The results obtained by this experiment showed a general decrease in the model performance, obtaining an average record-wise AUC-ROC of 59%, versus the previously obtained 65%. However, the most interesting result was the significative decrease in the individual-wise performance, passing from the original 81% AUC to a 61%. This outcome could indicate that the model is strongly affected by the number of records per individual, since the results on the rebalanced data showed a mild difference between the record- and the individual-wise accuracy.

## 2.3 ECG Classification

The last case of study of our work was the classification of ECG signals for diagnostic purpose. Till now, both in the HAR task and in the Parkinson's Disease detection, data were provided by accelerometer and gyroscope sensors, therefore we were interested in testing our models also on other types of sequential data, that could have direct biomedical applications. The PTB-XL [20] is a recently created ECG dataset, representing the largest collection of well-annotated ECG signals. Different Deep Learning algorithms were already tested on the PTB-XL, as described in [17]. As reported in the results, the best performing methods were once again Convolutional Neural Networks, in particular variants of the 1D-ResNet and 1D-Inception Network.

Therefore, the previously evaluated 1D-ResNet model was tested on the classification task of the 5 diagnostical superclasses. This problem is slightly different form the previously seen, since pathologies are not mutually exclusive, therefore representing a multi-label classification problem. For this reason, the architecture of the baseline network needed to be slightly changed, in order to handle this type of output.

### 2.3.1 Multihead ResNet Architecture

To permit the multi-output classification instead of having only one final fully-connected layer with single output unit, five different dense layers were connected to the 1D-ResNet backbone, each one with a single output unit. The proposed architecture is shown in Figure 2.19. The number of outputs is equal to the number of classes to predict; therefore, a sigmoidal activation was applied to perform a binary classification for each pathology. The loss function was computed as the sum of each single output loss:

$$Loss = \sum_{i=1}^{n} Loss^{i}_{binary}$$

where $Loss^{i}_{binary}$ is the Binary Cross-Entropy loss computed for class $i$.

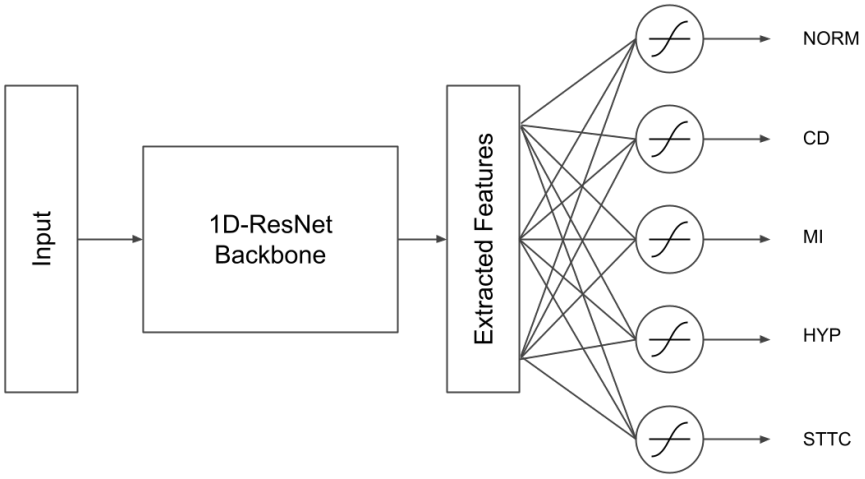


Figure 2.19: Architecture of the multihead 1D-ResNet trained for ECG classification.

### 2.3.2 Multi-ResNet Architecture

With the aim of further improving the classification accuracy another architecture was created. This time, only the input layer was shared and a different ResNet model, with single sigmoidal output unit, was trained for each one of the classes. This permits to learn separate networks that are highly specialized in recognizing only one pathology. Therefore, every single backbone could be potentially changed in order to find the best performing tuning for each class. In our experiments the only applied modification was doubling the number of filters in the convolutional layers of the HYP-specific ResNet, since it was the most difficult class to predict.

The structure of the complete model is shown in Figure 2.20. As the previous architecture, the total loss is computed as the sum of each Binary Cross-Entropy loss computed for each output unit.

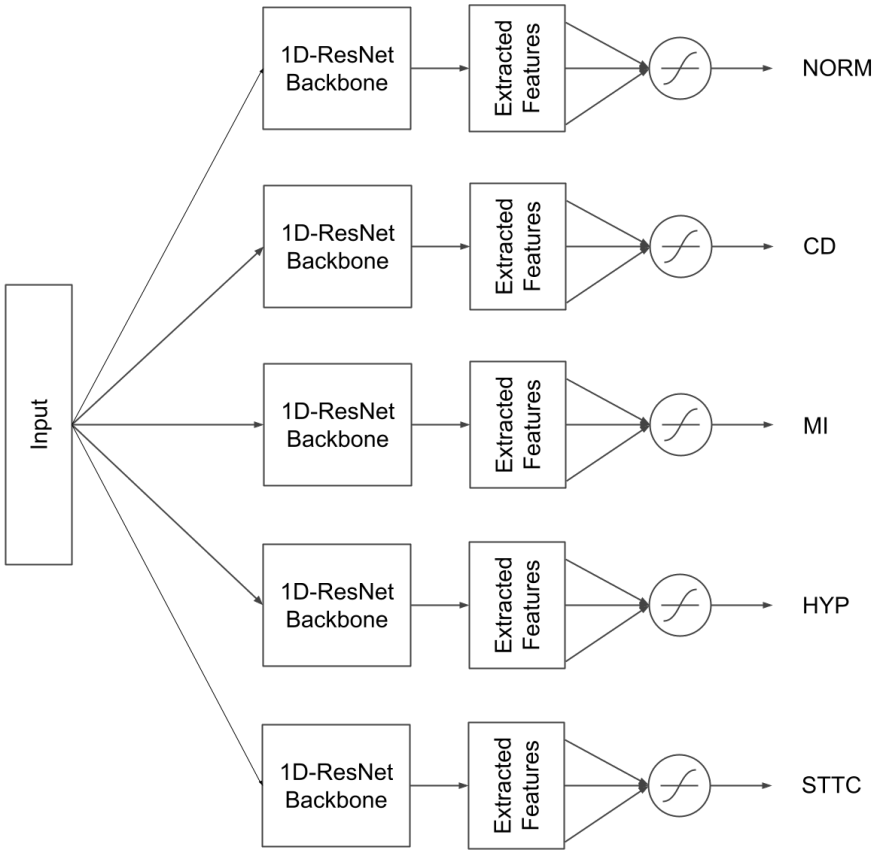


Figure 2.20: Architecture of the second ResNet-based architecture trained for ECG classification.

Despite potentially incrementing the discriminative power of the model, this architecture has the evident problem of consistently increasing the overall computational cost, that grows proportionally to the number of classes to predict. For this reason, this model has a practical use only when dealing with the classification of few groups.

### 2.3.3 Experimental Results

For our experiments one of the precomputed folds of the PTB-XL, suggested by the authors, was selected as the testing set. The remaining data was randomly divided in 70% training and %30 testing sets. As for the Parkinson's Detection task, the 1D-ResNet showed a long computational time, therefore it was impractical to perform a proper evaluation procedure, such as a cross-validation, considering the limitations imposed by the Google Colab environment. Both the proposed architecture were trained for 20 epochs, and the best performing weights settings encountered during the validation process were selected, and finally evaluated on the chosen testing set. In addition to the ROC-AUC score, computed in [17] and used as the main value to compare the models results, we decided to calculate further evaluation metrics in order to obtain a more descriptive picture of the actual performances. In particular, the Precision-Recall curve and its relative AUC score was computed in order to take consideration of the class imbalances. In this way we obtained a score that was more reflective of the actual performance of the tested models in the specific experiment, instead of having a general measure of their classification power in a hypothetical balanced scenario.

In Figure 2.21 are compared the loss values and the PC-AUC scores computed for each separate class over the 20 training epochs, considering both the architecture. As we can see, in both the cases, the most accurate predictions were made for the Normal ECG signals, with a validation accuracy around the 90%, while the worst predictions are produced for the signals reflecting Cardiac Hypertrophy, with validation accuracies around the 70%.

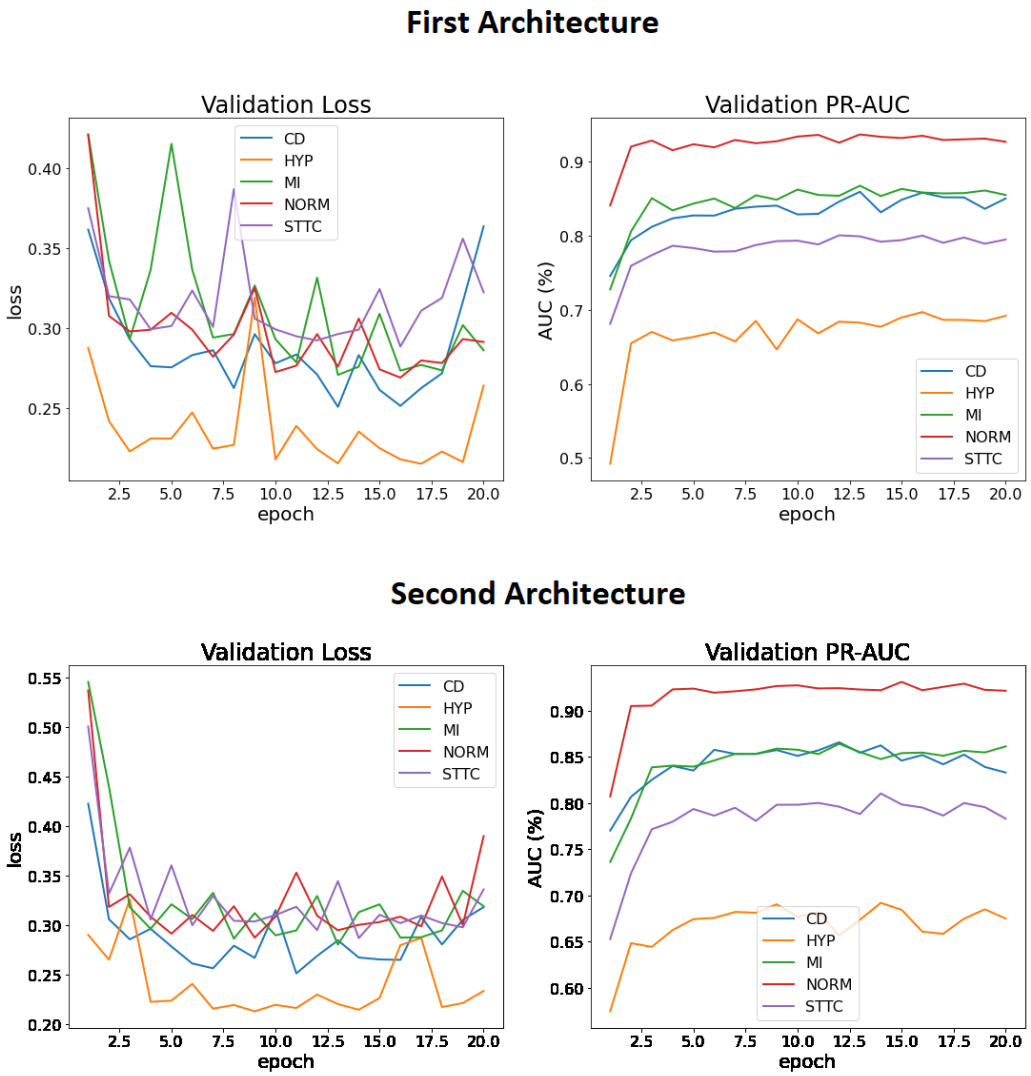


Figure 2.21: Validation results in terms of loss function value and PR-AUC score obtained by the two different architectures during the 20 epochs.

In addition, a set of comprehensive evaluation scores were computed by averaging the single values obtained for each specific class. In Table 2.2 are reported the obtained scores both in the validation and testing procedures, and they are relatable with the ones provided in [17] (ROC-AUC 92.8%). For the evaluation on the testing set the PC curves were plotted for both the two models and are shown in Figure 2.22.

| | 1st ResNet-based architecture | | 2nd ResNet-based architecture | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| **Avg. PR-AUC** | 82.9 | 80.8 | 82.7 | 80.3 |
| **Avg. ROC-AUC** | 93.4 | 92.3 | 93.1 | 91.8 |
| **Avg. Accuracy** | 89.3 | 88.8 | 88.8 | 88.2 |

Table 2.2: Summary of the validation and training results obtained by the two models

In general, the two proposed architecture provided very comparable results, probably implying that the features extracted from each ResNet backbone in the second model are very similar to the overall features extracted by the single ResNet in the first architecture. Therefore ,our last solution wasn't able to learn any additional class-specific characteristics, even for the HYP disease, in which its backbones filters were doubled.

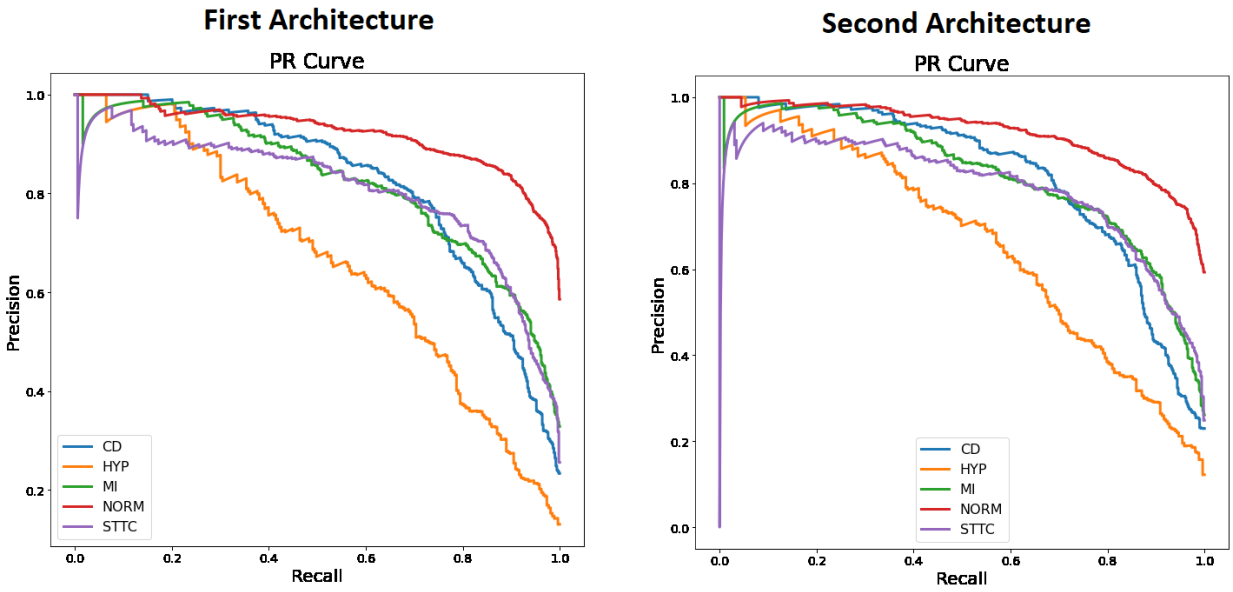


Figure 2.22: Precision-Recall curves obtained by the two models on the testing data for each class.

A further tuning of every single subnetwork present in the second architecture is needed to better discriminate the data, and learn different specific features with respect to the baseline model.

# 3 Discussion

The comparison between all the results obtained by our experiments composed a good general overview of the signal classification problem. In the following section, a discussion is made. Focusing on the results of each tested model, the discussion highlights the critical points and the possible implications.

## 3.1 Human Activity Recognition

The results obtained by the tested model in the HAR task are summarized in the following table:

| | **SCUDO** | **SVM** | **HMME** | **Improved HMME** | **CNN** | **ResNet** |
|---|---|---|---|---|---|---|
| **Accuracy (%)** | 41.7 | 89.5 | 76.5 | 90.0 | 91.0 | **94.0** |

The deep learning approach appeared to be the most promising solution. This conclusion is supported by the literature and reflects the trend of recent years, with deep learning models that generally outperform traditional machine learning methods when dealing with complex data and a sufficiently large number of samples.

Our experiments with SCUDO showed its limitations when dealing with this type of data. Indeed, SCUDO is originally designed for the classification of gene expression profiles, in which we typically have few highly dimensional samples, both in the control and test groups. Since the algorithm creates a graph with the number of nodes equal to the number of samples, when the training set is too large the computational cost becomes unsustainable, due to the distance computation between each couple of records

For this reason, it is necessary to train SCUDO with a quite small subset of data, learning a less complex model that can result in an insufficient predictive power. Moreover, from the graphs produced during our experiments, we can deduct that SCUDO was unable to make a good separation between classes that present a small variation in the features' value. In other words, SCUDO was not able to perform a satisfying stratification of the activities that share similar characteristics, such as walking, walking upstairs and walking downstairs, or sitting, standing and laying.

SVM instead resulted in a noticeably good performance, both in terms of classification accuracy and computational cost, showing comparable results to the ones reported in the literature [14]. This implies that the precomputed set of 561 features is a good representation of the data sequences. Despite this, it is generally preferable to rely on a model that autonomously extracts features that better represent the data (as we have seen for the CNN models). This is because the traditional way of extracting features is based on human assumptions that may lead to producing redundant information or even missing some important discriminative features that are difficult to extract intuitively. Another relevant issue (as previously discussed) is the loss of the temporal information that is implicitly encoded in the time sequence data. The set of extracted features represents some characteristics of the entire sequences, but they are not able to capture the internal variations between sequential parts of the signal..

The HMME model was revealed to be a good compromise between feature extraction and the modelling of sequential data. In this solution, the extraction of features is not performed on entire sequences, but only on subparts of them. This acts as a subsampling technique to reduce the sequence length, but at the same time, it preserves some of the intrinsic temporal information. Thanks to our improvement on the original model, we were able to reach satisfactory results, obtaining a slightly better classification performance than the one obtained with SVM. The main problem with this method was the lengthy period required to fit the model. For this reason, it became impractical to test it on the two biomedical cases of study without applying drastic approximations, considering that we were dealing with larger datasets composed by longer sequences (1000 reads for the PBL-XL, and 4000 reads for the mPower dataset, versus the 128 reads of the UCI HAR dataset).

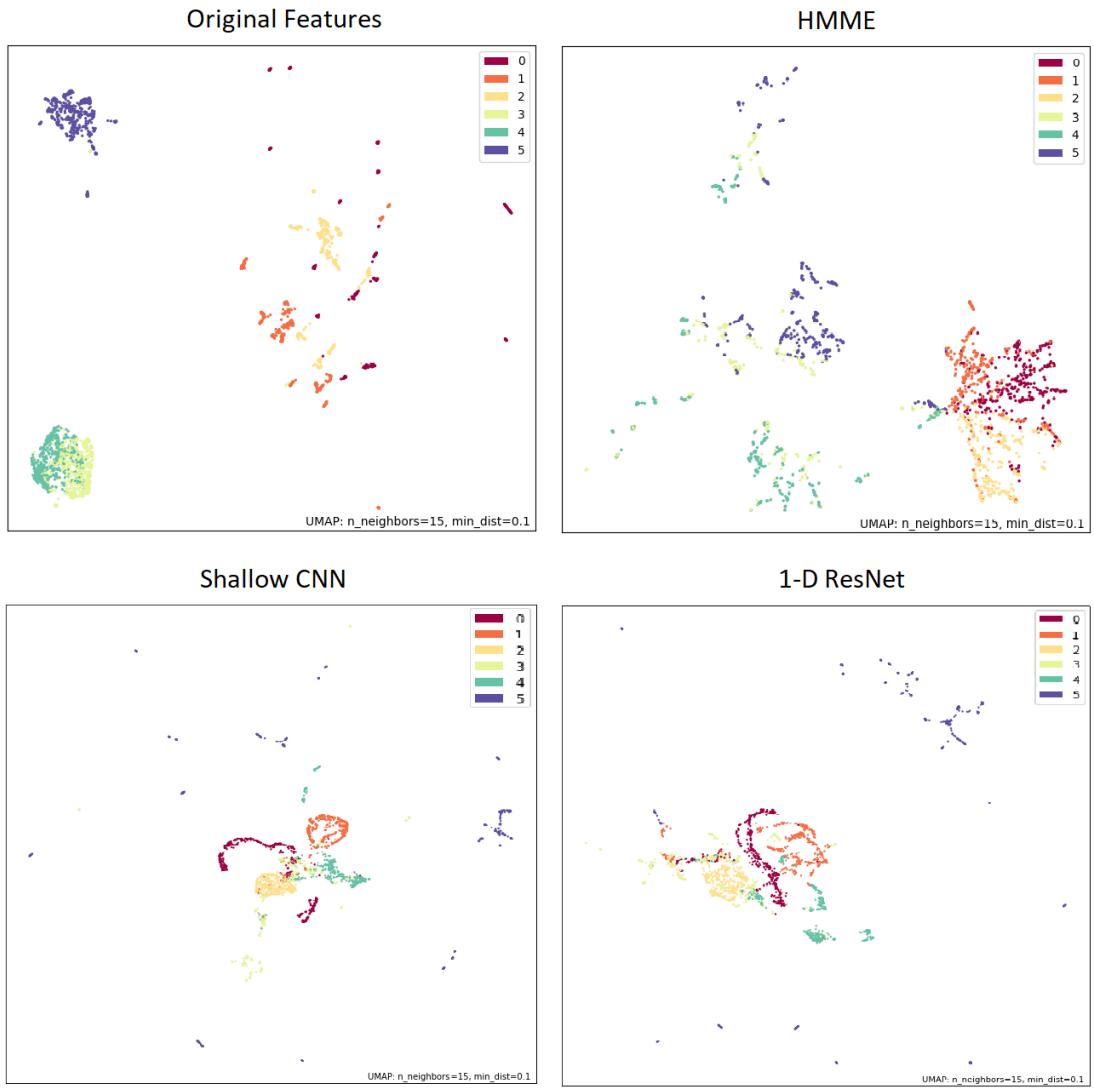


Figure 3.1: 2-dimensional representation of the various features extracted from the models. The labels, from 0 to 5, represent respectively the classes walking, walking upstairs, walking downstairs, sitting, standing and laying.

As shown in the results, the Deep Learning approach was the one with the highest performance. 1D-ResNet architectures were revealed to be a great solution for the classification of the sequential data. The key factor leading to outstanding results was the CNN's ability to learn optimal features that can successfully discriminate input data. In Figure 3.1 a 2-dimensional graphical representation of the features is shown for each model, comprehending the set of pre-computed features provided in the UCI HAR dataset. These graphs are computed applying UMAP [12], a dimensionality reduction method similar to t-SNE, but with the ability to better preserve the global structure of the data. As we notice from the picture, in general, the features extracted by the CNN models show a better separation of the classes.

As shown in the results, the Deep Learning approach was the one with the highest performance, in particular 1D-ResNet architectures revealed to be a great solution for the classification of the sequential data. The key factor leading to outstanding results was the CNNs ability of learning optimal features that can successfully discriminate input data. In Figure 3.1 a 2-dimensional graphical representation of the features is shown for each model, comprehending the set of pre-computed features provided in the UCI HAR dataset. These graphs are computed applying UMAP [12], a dimensionality reduction method similar to t-SNE, but with the ability of better preserving the global structure of the data.

As we can notice from the picture, in general the features extracted by the CNN models shown a better separation of the classes.

## 3.2 Parkinson's Diseases Detection

For the Parkinson's Disease Detection, we were successfully able to replicate the results obtained in [25]. As we pointed out in the previous chapter, the drop in the record-wise accuracy obtained by the model, with respect to the individual-wise, can be reflective of the difficulty in extracting from each walking sequence reliable markers representing the two conditions. Additional types of models were further tested, combing the 1-D ResNet architecture with different classification algorithms. These experiments led to an increase of the record-wise classification performance, with the drawback of drastically affecting the computational cost of the training procedure. Despite our obtained results, a further evaluation of the models is needed, since for computational impediments (Google Colab limitations in terms of runtime and memory usage) it was impractical to build a more complete evaluation pipeline, such as cross-validation.

In our opinion, the main problem encountered while dealing with the Parkinson's Detection task was the mPower dataset structure. In principle, it needs to be considered that the walking experiments performed by the volunteers during the data collection were autonomously made, without any supervision. The absence of a supervised experimental protocol has probably led to recording unreliable signals. Moreover, some of the personal information could have been incorrectly self-reported, causing a wrong data annotation. Considering these factors, and the different biases and imbalances present in the dataset, it becomes problematic to build a reliable model able to successfully fit the data. This highlights the difficulty of analysing crowd-sourced data, often characterized by noisy and unreliable records. Moreover, we could also hypothesize that devices such as smartwatches should be preferable with respect to smartphones when collecting the data in an unsupervised manner since they could be less prone to experimental variations (e.g., the position of the smartphone in the pocket, or the position of the pocket with respect to the subject's body). In the future, it could be beneficial to conduct another experimental collection of data in a more structured and supervised manner, in order to build a satisfactory model that can be subsequently tested on a crowd-sourced experiment.

However, even if we didn't obtain significant improvements (in terms of individual classification accuracy with respect to the model proposed in [25]) our replication provides a new full-Python implementation, relying on up-to-date libraries that significantly improved the computational time. Moreover, in our experiments we proposed some more complex alternatives to the original CNN baseline, composing the model ensemble. These solutions should be further explored in a more in-depth analysis.

## 3.3 ECG Signals Classification

In the last case of the study, we tested our methods on the PTB-XL ECG collected signals, in order to evaluate their performances when dealing with other types of data, different from the acceleration and rotation signals. For our experiments, the 1D-ResNet baseline was modified in two different architectures, with the aim of computing the multi-label predictions. The two different ResNet-based models gave similarly good results for the classification of pathologies, comparable to the ones published in [17].

Despite our measured performances reflecting the reported results, we computed further appropriate evaluation metrics to obtain a better evaluation of the models. Indeed, in [17] the mainly considered score for the models' evaluation was the macro-averaged ROC-AUC. Despite being in general a good indicator of classifier performance, it tends to give a too optimistic picture of the model when class imbalance is present. This is the case of the considered ECG multi-label classification problem, in which metrics are computed separately for each label in a one-vs-all manner, and the final score is computed as the overall average. For this reason, in addition to the ROC-AUC, we computed the PR curve and its relative AUC score, that in general provide a better understanding of the actual model performance when moderate to severe class imbalance is present. The combination of these two AUC metrics gave a more detailed estimation of the actual classifiers' discriminative power.

# 4 Conclusions

In this thesis are reported the experiments and the corresponding results obtained during months of research. In this work, we provide a general overview of the main approaches and techniques to deal with signals data, recorded by wearable sensors. We wandered from the more traditional machine learning pipelines, based on feature extraction, to the more recent deep learning architectures and hybrid methods.

A general analysis was computed on a well-known problem, the Human Activity Recognition, comparing to each other the results obtained by the tested models and the ones reported in the corresponding literature.

Artificial Neural Network-based models, 1-dimensional Convolutional Neural Networks, were revealed to be the most promising and versatile solution when dealing with time sequence classification problems. The measured performances for the tested deep learning-based solutions reflected the general trend observed in recent years, in which we see ANN models usually outperforming traditional machine learning techniques when trained on large and structured datasets.

Our subsequent experiments on the Parkinson's Diseases Detection and ECG classification provided good starting point solutions for analysing data from sensors in a biomedical scenario. With our work, we explored the field of digital biomarkers research, with the aim of performing preliminary disease diagnoses. For these particular tasks we tested CNN-based methods, and furthermore, some hybrid models, combining the artificial networks for the feature extraction with traditional Machine Learning classifiers to perform the final predictions.

The results obtained from our methods were compared to the literature, and moreover, we highlighted the most critical and problematic aspects of the involved data. In the experiments on the Parkinson's Disease Detection task, we experienced the difficulty of analysing noisy and potentially unreliable crowd-sourced data. Therefore, this thesis could be considered as a strong starting point for a more in-depth and comprehensive analysis in the field of sequential data classification for a variety of biomedical applications.

## 4.1 Future Work and Possible Improvements

The main issue in researching new models to be applied for biomedical purposes is the lack of large well-annotated and quality datasets. Indeed, especially when training Artificial Neural Networks, a substantial amount of data is needed to fit satisfactory models. A virtuous example of such a type of dataset is the PTB-XL, described in Chapter 1.1.3, representing the first large collection of annotated ECG signals, comprehending disease-related information for a training model with a diagnostical purpose.

Therefore, the main aspect to focus on in the feature should be the collection of new data, following commonly shared protocols. The ideal scenario would be obtaining a large dataset comprehending different signals extracted by a multimodal device and involving many control individuals and patients presenting different diseases. This utopic project could involve different institutes and clinics, with the aim of collecting data in a professionally supervised manner.

The methods proposed in this thesis could be tested on different types of signals, with the aim of creating a model that can successfully handle multimodal data, in order to perform more accurate diagnostical predictions. Examples could be the blood pressure and the Electroencephalogram.

Additionally, with the increase of the considered data channels and the complexity of the chosen models, also comes an increment of the computational cost.

In our work we relied on the hardware resources provided by the Google Colab environment to speed up the training process, however, we have also faced its limitations when dealing with more complex models and higher-dimensional data. For this reason, it would be necessary to rely on more powerful tools in the feature, such as the University GPU cluster.

# 5 Supplemental Materials

All the source code for replication is publicly available in a GitHub repository at the following link: https://github.com/gabrieleberrera/QCB_sensors_analysis_2021

# Bibliography


[1] Iso central secretary. health informatics – standard communication protocol – part 91064: Computer-assisted electrocardiography. *Standard ISO 11073-91064:2009, International Organization for Standardization, Geneva, CH*, 2009.

[2] Himani Bhavsar and Mahesh H Panchal. A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(10):185–189, 2012.

[3] Brian M Bot, Christine Suver, Elias Chaibub Neto, Michael Kellen, Arno Klein, Christopher Bare, Megan Doerr, Abhishek Pratap, John Wilbanks, E Dorsey, et al. The mpower study, parkinson disease mobile data collected using researchkit. *Scientific data*, 3(1):1–9, 2016.

[4] Matteo Ciciani, Thomas Cantore, and Mario Lauria. rscudo: an r package for classification of molecular profiles using rank-based signatures. *Bioinformatics*, 36(13):4095–4096, 2020.

[5] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Yong-Joong Kim, Bong-Nam Kang, and Daijin Kim. Hidden markov model ensemble for activity recognition using tri-axis accelerometer. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3036–3041. IEEE, 2015.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Mario Lauria. Rank-based transcriptional signatures: a novel approach to diagnostic biomarker definition and analysis. *Systems Biomedicine*, 1(4):228–239, 2013.

[10] Mario Lauria, Petros Moyseos, and Corrado Priami. Scudo: a tool for signature-based clustering of expression profiles. *Nucleic acids research*, 43(W1):W188–W192, 2015.

[11] Mark Lew. Overview of parkinson's disease. *Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy*, 27(12P2):155S–160S, 2007.

[12] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[13] Bolu Oluwalade, Sunil Neela, Judy Wawira, Tobiloba Adejumo, and Saptarshi Purkayastha. Human activity recognition using deep learning models on smartphones and smartwatches sensor data. *arXiv preprint arXiv:2103.03836*, 2021.

[14] Jorge-Luis Reyes-Ortiz, Davide Anguita, Alessandro Ghio, and X Parra. Human activity recognition using smartphones data set. *UCI Machine Learning Repository; University of California, Irvine, School of Information and Computer Sciences: Irvine, CA, USA*, 2012.

[15] Benjamin Schuster-Böckler and Alex Bateman. An introduction to hidden markov models. *Current protocols in bioinformatics*, 18(1):A–3A, 2007.

[16] Solveig K Sieberts, Jennifer Schaff, Marlena Duda, Bálint Ármin Pataki, Ming Sun, Phil Snyder, Jean-Francois Daneault, Federico Parisi, Gianluca Costante, Udi Rubin, et al. Crowdsourcing digital health measures to predict parkinson's disease severity: the parkinson's disease digital biomarker dream challenge. *NPJ digital medicine*, 4(1):1–12, 2021.

[17] Nils Strodthoff, Patrick Wagner, Tobias Schaeffter, and Wojciech Samek. Deep learning for ecg analysis: Benchmarks and insights from ptb-xl. *IEEE Journal of Biomedical and Health Informatics*, 25(5):1519–1528, 2020.

[18] Ramasubramanian H Sundaram. The baum-welch algorithm. Technical report, Technical Report, 2000.

[19] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[20] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):1–15, 2020.

[21] Shuhui Wang, Jiawei Xiang, Yongteng Zhong, and Yuqing Zhou. Convolutional neural network-based hidden markov models for rolling element bearing fault identification. *Knowledge-Based Systems*, 144:65–76, 2018.

[22] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[23] Guorong Xuan, Wei Zhang, and Peiqi Chai. Em algorithms of gaussian mixture model and hidden markov model. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 145–148. IEEE, 2001.

[24] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[25] Hanrui Zhang, Kaiwen Deng, Hongyang Li, Roger L Albin, and Yuanfang Guan. Deep learning identifies digital biomarkers for self-reported parkinson's disease. *Patterns*, 1(3):100042, 2020.