POLITECNICO DI TORINO

Master's Degree in Biomedical Engineering

Master's Degree Thesis

Human Activity Recognition on smartphone using Convolutional Neural Networks

Supervisors
Prof. Gabriella Olmo
Ing. Luigi Borzì

Candidate
Giorgia Mondello

Academic Year 2020/2021

Summary

Human activity recognition (HAR) is a key research area in Human Computer Interaction (HCI) that is grown and spread enormously in recent years thanks to the widesprade usage of portable and wearable devices, as smartphones. The mainly aim of HAR is to identify a variety of daily activities (ADLs) performing by an individual at a given moment, to maintain healthy lifestyle, help patient rehabilitation and to detect and diagnose automatically precocious illnesses, such as the Parkinson's disease. HAR relies on two approaches: vision-based HAR and sensor-based HAR. The approach used in the current work was sensor-based HAR, that enables to collect data extracted from two type of wearable sensors, namely accelerometer and gyroscope, which are embedded into a smartphone.

Inertial sensors data was acquired by three public available datasets: UCI HAR, HAPT and RealWorld HAR dataset, containing kinematic data of human subjects that perform some dinamic (walking, sit-to-stand, stand-to-sit) and postural activities (sitting, standing) following a specific protocol. Each dataset was splitted into training set (80% of the total data) and a test set (20% of the total data). Then, the *Hold out* technique was applied on the training set, in order to divide the training set into the train set (80% of the training set) and the validation set (20% of the training set). In addition, Leave-One-Subject-Out (LOSO) was applied in RealWorld HAR to improve generalization and avoid over-fitting. In this work, the proposed model to make prediction and identify classes for new data was the Convolutional Neural Network (CNN), that is a Deep Learning Network that learns directly from data, eliminating the need to manually extract features. The model was built using the Keras deep learning library and developed on five different architectures having different combinations of Convolutional layers and Max-pooling layers.

The model was optimized by tuning two hyper-parameters, that are the number of filters and the number of filter size in order to observe their effects on the performances. Finally, the best model configuration was selected for each database.

The optimized model was trained on the training set, evaluated on the validation set and used to make prediction on the test set of each database. In addition, cross-corpus HAR was performed to improve generalization: the CNN model with the five architectures was tested on new test data of two databases using labeled training data of the other dataset.

For each architecture of each dataset, different performance evaluation metrics were computed: accuracy, specificity, precision, sensitivity and f_1-score. The model trained on the train set and tested on the own test set led good performances: 98.5% of accuracy in HAPT dataset, 89.3% of accuracy in UCI HAR dataset and 84.9% of accuracy in RealWorld HAR dataset.

On the other hand, performances obtained from cross-corpus test led worst performances. The performance obtained by the CNN model confirm the ability of the CNN model to make correct prediction on test set of each database and suggest how the model is not reliable to make prediction on new test data using labeled training data.

Contents

List of Tables

List of Figures

Abbreviations

AAL	Ambient Assisted Living
ADLs	Activities of Daily Living
CNN	Convolutional Neural Network
DL	Deep Learning
DFN	Deeply Connected Network
DT	Decision Trees
FN	False Negatives
FP	False Positives
HAPT	Human Activity Postural Transitions
HAR	Human Activity Recognition
HCI	Human Computer Interaction
IoT	Internet of Things
IMUs	Inertial Measurement Units
k-NN	k-Nearest Neighbours
LOSO	Leave-One-Subject-Out
LSTM	Long Short-Term Memory
ML	Machine Learning
MHEALTH	Mobile HEALTH
NB	Naïve Bayes
PD	Parkinson's Disease
RF	Random Forests
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
RBM	Restricted Boltzmann Machine
SAE	Stacked Autoencoder
SVM	Support Vector Machine
TP	True Positives
TN	True Negatives
UCI	University of California Irvine
WISDM	Wireless Sensor Data Mining

Chapter 1

Introduction

In the last decade, sensor technology has developed considerably and has had a socio-cultural impact in everyday people's life. The advancement has concerned multiple points of views, including power, size, manufacturing costs, portability, storage capability and power of sensing. These improvements have enabled a wide range of sensors to be integrated into portable and wearable smart devices in order to create them more useful for people's health.
The rapid spread of portable and wearable devices led Human Activity Recognition (HAR) to grow enormously in recent years in a variety of applications of different knowledge areas, including healthcare, Internet of Things (IoT), security, homecare, communication, environmental monitoring, transportation, sports[1], [2].
HAR is a key research area in Human Computer Interaction (HCI) which enables continuous monitoring of human behaviour, support on patients rehabilitation, surveillance in smart-home environments, precocious diagnose of illnesses [25], not only in ambient assisted living (AAL), but also in uncontrolled settings.
HAR typically relies on two approaches: vision-based HAR and sensor-based HAR [1]. Vision-based HAR data analyzes images or videos through optical sensors. Though video cameras are widely used by researchers, collecting video data presents several issues concerning privacy limitations and high computational resources [4],[44]. In addition this approach relies on image resolution, illuminations change and other factors that can influence image quality [28].
Sensor-based HAR is the most common approach used in real world applications to collect raw data extracted from sensors embedded into smartphones or other wearable devices. Nowadays, smartphones are considered one of the most important part of sensor-based HAR systems and contain a wide set of embedded sensors, such as location, motion and direction indicators, that collect, process raw data and monitor continuously activities of daily living (ADLs) in the area of AAL [16].
Sensor-based HAR can be intended as a classification issue [53], that analyzes data acquired from various types of sensing devices and identifies a variety of daily activities performing by an individual at a given moment, to maintain healthy lifestyle.
The human activities are defined as a set of repeated actions over time and can be divided into two distinguishable sets: simple activities and complex activities. Sample activities include all those activities which are repeated daily, as walking, sitting, standing, running, jumping, while complex activities include activities as eating, sleeping, driving, smoking [5]. The most common sensors types used in sensor-based HAR are accelerometer, gyroscope and magnetometer [44]. The accelerometer detects acceleration (m/s^2) and tilt to determine movement and exact orientation along the three axes, while the gyroscope provides orientation, direction and rotation details and records angular velocity (rad/s).
These two wearable sensors offer many advantages. In more detail, advantages of accelerometers include high sensitivity, high impedance, high frequency response. In addition accelerometers

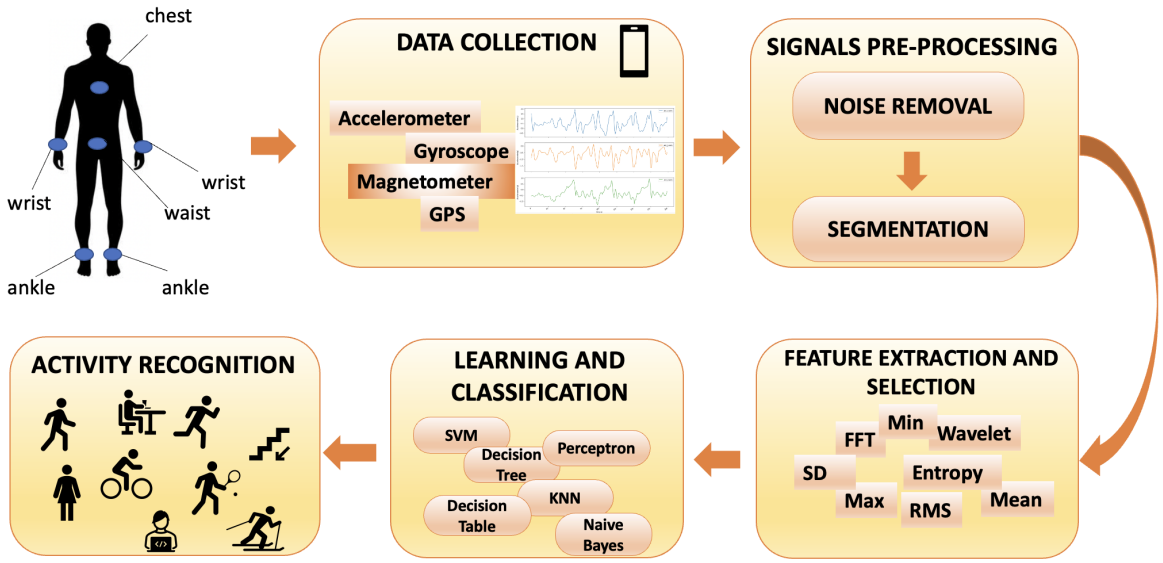

Figure 1.1: Set of steps used by machine learning algorithms in HAR applications.

are cheap and durable over time, while gyroscopes are fast and have an higher resolution. However, these sensors show also some drawbacks, concerning accelerometer's efficiency which tends to decrease over time and the gyroscope' s dependence on the rotation of the earth [1]. In terms of algorithmic implementation, HAR is mainly based on Machine Learning (ML) and Deep Learning (DL) algorithms. Some of the most common ML algorithms are Naïve Bayes (NB), k-Means Clustering, Support Vector Machine (SVM), Linear Regression, while some of the most common DL algorithms are Convolutional Neural Network (CNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs). The differences between the two approaches concern especially the way in which the features are extracted, manually in ML and automatically in DL algorithms. Feature extraction is a crucial step tp determine performances on activity recognition systems.

ML algorithms present many drawbacks on HAR applications: they require many pre-processing data steps, that consist of segmentation, in which data is divided into overlapping or non-overlapping windows and feature extraction; they haven't universal procedures for selecting appropriate features, but extract manually the features via a heuristic way [32] based on human experience and knowledge, that may not be able to select the optimal set of features. Additionally, feature extraction takes a long time and it may be necessary to apply some methods to reduce the dimensionality of the data, such as Principal Component analysis (PCA) [25], in order to remove irrelevant features and improve the accuracy of the models.
Only superficial characteristics can be learned with ML algorithms and this renders such methods unable to recognize complex activities.
In addition, ML approaches require a large amount of labeled data to train the model, but most of the activity data are unlabeled in real applications [32]. Another disadvantage of traditional methods is that data from the same person is present in both the training set and the testing set [48] and therefore the methods may not be able to generalize new activities and users.
Figure 1.1 summarizes the set of steps used by the ML algorithms. First, raw data signals are collected by smartphones or other wearable devices integrated with inertial sensors, as accelerometer, gyroscope, magnetometer, GPS, which record inertial signals at a constant frequency rate, The smartphone can be placed in different positions on the user's body, such as chest, waist, ankle, wrists. Choosing the best smartphone location is important in determining data quality and [53] classification performance. Furthermore, the orientation of the smartphone must be chosen carefully, as it could affect the accuracy of the classification models.

The second step is data preprocessing, including noise removal and segmentation.
Signal preprocessing tends to reduce noise by applying a series of filters. The most commonly used filters are the low pass filters [18] Butterworth [19], Kalman [20] and Moving Average [21]. Segmentation is meant to separate the data into segments of equal length to analyze them separately and sequentially. The most common technique used for segmentation is Sliding Window [27]. The third step relates to feature extraction and feature selection. In ML algorithms the characteristics are extracted manually and are mainly based on two main characteristics of the domain: time domain and frequency domain [6]. The time domain extracts statistical information from the signals, such as minimum, maximum, peak amplitude of the signals, while the frequency domain obtains functions based on the frequency spectrum of a certain time window, such as energy, power, entropy and represent the relative characteristics the periodicity of the human activities. After the data processing, segmentation, and feature extraction stages, the next stage is learning and classification, where the ML models are trained on the train set and evaluated on the test set in terms of recognition metrics. activities. Commonly used assessment metrics in smartphone-based HAR literature are: accuracy, sensitivity, specificity, precision, recall and f-measure [27].
To solve the limitations of the ML algorithms described above, DL methods are the most common methods used in HAR applications, as they do not require signal preprocessing and the feature extraction occurs automatically during the training process of classification models in the raw data, as shown in the **Figure 1.2**.
The next step consists in the construction of models (CNN, RRN ..) used for the classification of human activities. Deep learning and recognition of human activities have been progressive areas and a good number of surveys and reviews have been published in recent years, as reported in the 2 chapter. The main contribution of this work is to show how DL methods and in particular the CNN model can be used for the recognition of human activities through data collected by inertial sensors embedded in a smartphone.
The remainder of this work is organized as follows: in the **Chapter 2** some recent works for HAR applications are presented. The **Chapter 3** describes the datasets employed, the data pre-processing, the classification model and the description of the performance evaluation metrics used in this study. The **Chapter 4** presents and analyzes results obtained from the training process and model evaluation. The results obtained are discussed in the **Chapter 5**. Finally in the **Chapter 6**, conclusions are drawn and further improvements are proposed.

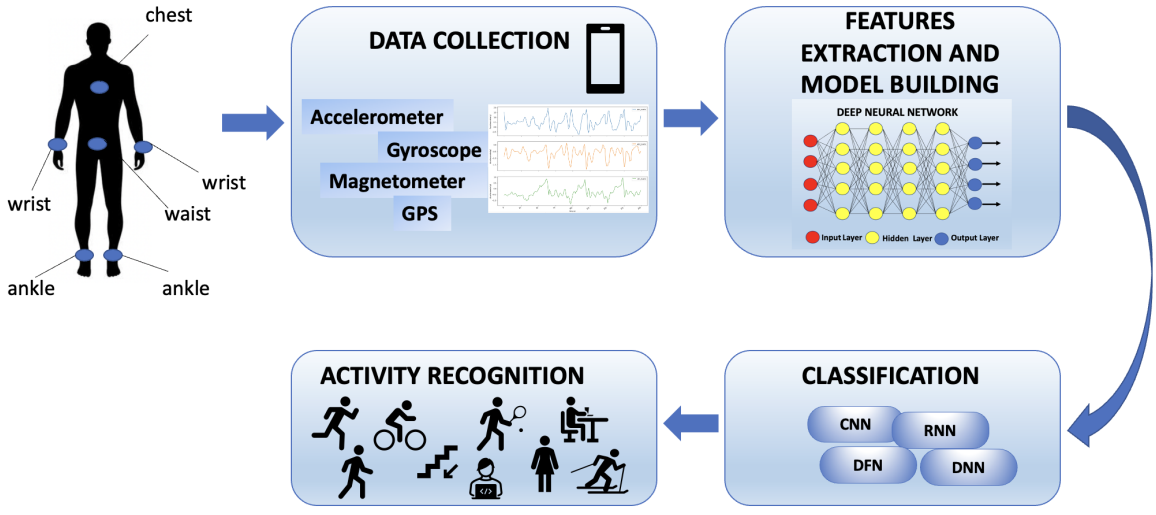

Figure 1.2: Set of steps used by deep learning algorithms in HAR applications.

Chapter 2

Related works

Several studies focusing on Human Activity Recognition Analysis (HAR) have been published in recent years, including sensor-based HAR and video-based HAR.
Although video cameras have been widely used in HAR applications, video data collection has many privacy issues [1], their elaboration requires high computing resources [3] and their performance depends on lighting, visual angle and other factors [28]. For this reason, studies frequently rely on the use of wearable sensors, in particular on the use of the smartphone as an acquisition device for the recognition of daily activities (ADL) [4], [5], [**?**]. Nowadays, the smartphone is very popular and can be considered the perfect tool for the analysis of human activities due to its ability to collect and process data, transmit and receive data and connect with other wearable sensors [24]. It also reduces costs and overcomes the problem of learning another type of device [4]. Choosing the location of the smartphone is crucial, as different locations produce different signals and require different analyzes to recognize the same activity. Many studies have shown that the waist is the best position for recognizing simple activities [12],[13],[14],[15],[17],[18],[22].
Generally, the smartphone is integrated with inertial measurement units (IMU) which contain accelerometer, gyroscope, magnetometer, GPS. The accelerometer and gyroscope combination has been used primarily for HAR [29]applications and has shown excellent results in human motion analysis and activity monitoring [25].
For a long time, Machine Learning (ML) algorithms have been applied in HAR applications achieving remarkable performance [1]. In fact, several works have been based on the traditional methodology [33],[34], [35], [36],[37],[38], using algorithms such as: Naïve Bayes (NB), Support Vector Machine (SVM), Linear Regression, Logistic Regression, Random Forests (RF), Decision Trees (DT) and k-Nearest Neighbors (k-NN).
However, conventional methods require many preprocessing steps, including noise removal, segmentation, normalization, and manual features extraction [32], which relies on human experience and knowledge. Studies have shown that it is very difficult to measure efficient performance of manually engineered features across different applications. Plus the selection of handcrafted features takes time [31], dimensionality reduction and is often arbitrary [25]. Furthermore, conventional approaches often require a large amount of well-labeled data to train the model and most of the activity data remains unlabeled in real applications [32].
On the other hand, the last decade has seen significant growth in deep learning (DL) algorithms, thanks to their superior performance in several real-world problems. Unlike the traditional ML methodology, the DL methods have modified in particular the procedures used for the extraction and selection phases of the characteristics. Starting from raw input data, DL algorithms have the advantage of being able to automatically extract the optimal characteristics, without any human intervention. For this reason, several studies in recent years have relied on the deep learning approach in different real-world applications, such as security and health surveillance,

smart-home [1] and HAR applications [39],[40],[41],[42],[43]. In addition, DL algorithms work well with large datasets, which can be also unstructured datasets [1].
There are several DL methods used for real world applications, such as image processing or object detection. In the context of smartphone HAR, only five methods have been identified: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Deeply-connected network (DFN), Stacked Autoencoder (SAE) , and Restricted Boltzmann Machine (RBM) [24].
CNN is the most used DL model, which has been shown to achieve excellent results [46],[47], [11],[48], [57], [56].
In preparation for the neural network model, HAR sensor data must be pre-processed through a few steps, including normalization and segmentation. In the literature the most common methods for normalization are Min-Max and Z-score [3],[38], [49]. Regarding segmentation, the most common approach used in HAR tasks is the Sliding Window technique, in which the choice of window size is a crucial step in the HAR process.
Existing works have confirmed that small windows are used for simple activity detection. Conversely, large windows are normally used to capture the high variability of motion found in complex activities [50].
For the recognition of activity on motion sensors, window sizes are measured based on the time interval and frequency rate of data collection. Most datasets are sampled at 50 Hz, including the MHEALTH (Mobile Health) dataset [52] and the UCI HAR dataset [53], which are the most commonly used databases available in HAR fields. Considering a frequency of 50 Hz, studies have shown that the 1–2 s interval provides the best compromise between recognition speed and accuracy [51]. Studies generally consider time segments of size 128, which correspond to a duration of 2.56 s [53], [11], [54], [55]. Gholamiangonabadi *et al* [48] explored the impact of window size on accuracy. This work demonstrated that increasing the window size does not necessarily lead to an increase in accuracy. A large window, in fact, can lead to a decrease in accuracy. In this work, two Feed Forward Neural Network (FFNN) architectures and the CNN model were proposed and the MHEALTH dataset was selected. The sensors used were accelerometer, gyroscope and magnetometer, which were located on various parts of the body, such as chest, right wrist and left ankle. The first FFNN architecture consisted of 4 hidden layers, with 128 neurons in each hidden layer, while the second architecture consisted of 6 hidden layers. On the other hand, the first CNN architecture was composed of a convolution layer with 64 feature maps, a max-pooling layer and a fully connected layer with 32 neurons, two max-pooling layers and two fully connected layers with 64 and 32 neurons, respectively. Model evaluation was performed using Leave-One-Subject-Out Cross-Validation (LOSOCV) and 10-Fold Cross-Validation techniques. Experiments demonstrated that LOSOCV is a useful technique for estimating model performance, while k-fold cross-validation can lead to overestimation of accuracy. The CNN model with two convolutional layers and 1D filters achieved the highest accuracy of 99.85 % with traditional 10-fold cross-validation and of 85.1% with LOSOCV evaluation.
Ignatov *et al* [11] proposed a CNN model architecture, performed on WISDM [58] and UCI [53] dataset, containing accelerometer time series data obtained from smartphones. Data was collected from 36 and 30 different subjects while performing six activities: walking, jogging, climbing stairs, sitting, lying down, and standing. The CNN model was composed by a convolutional layer with 196 convolutional filters, followed by the ReLU function. Then a max-pooling layer was applied to reduce feature representation by 4 times. Then features passed to a fully-connected layer that consists of 1024 neurons and a dropout layer was applied with a dropout rate of 0.05 Finally, the outputs of the fully-connected layer passed to a softmax layer that computes probability distribution over six activity classes. In order to analyze how CNN structural parameters can influence classification results, the number and size of convolutional filters and the dropout rate were varied. Experiments demonstrated that smaller numbers of convolutional

filters do not give great results and n = 32 or n = 64 feature maps allow for similar accuracy of about 96 %. As for the filter size, the best accuracy was obtained with a filter size of 16, which allows to reach an accuracy of 96.67 %. A dropout rate between 0.04 and 0.1 proved to be the most efficient in this work. The proposed CNN model achieved high performance on both databases.

Peppas *et al* [23] proposed a CNN model performed on WISDM [58] and Actitracker [59] datasets, using only accelerometer data from a mobile device. The CNN model was composed of two identical blocks formed by a convolution layer and a max-pooling layer. The output of the max-pooling layer was then flattened and concatenated with the statistical features and passed to a fully connected layer consisting of 512 neurons. The ReLU function has been applied to its output. A dropout layer was added with a dropout rate of 0.5 to avoid overfitting. Finally, the output of the fully connected layer is passed to a softmax layer, which computes a probability distribution over six activity classes. Furthermore, this works provided an analysis of accuracy with varying window size and concluded that a window size of 50 provides efficient performance. Experiments demonstrated this architecture of the CNN model is better in terms of size, throughput, and performance and allows to achieve an accuracy of 94.18% on WISDM and an accuracy of 79.12 % on Actitracker, 0.5% and 2% higher than the previous state of the art.

Zebin *et al* [47] proposed a CNN model architecture for the classification of five ADLs using raw accelerometer and gyroscope data from a waist-mounted inertial sensor. The architecture was composed by four identical blocks composed by a convolutional and max-pooling layer. Then, the output of the max-pooling layer was flattened and passed to a fully-connected layer of 50 neurons. The ReLU function was applied to its output. A batch normalization layers and a dropout layer with a dropout rate of 0.2 were added to avoid overfitting. Finally, the output of the fully-connected layer passed to a softmax layer, which computes a probability distribution over five activity classes. The performances of the model were evaluated by varying the number of convolutional layers, the number of filters and the size of the kernel, observing their effect on detection speed and accuracy. Experimental results showed that with an increasing number of layers, the accuracy of the model increases starting from an accuracy of 82 % with a single layer and up to 94.9% with five levels. However, the complexity and execution time of the network increases due to the increasing number of parameters as the number of levels increases. By increasing the number of filters the model accuracy increases from 89 % with 6 filters up to 96.4 % with 50 filters. As the kernel size increases, model accuracy increases up to a kernel size of 12. Model accuracy deteriorates with larger filter sizes, such as 15 and 18. Furthermore, the work concluded that the batch normalized implementation makes the network to achieve stable training performance in almost four times fewer iterations.

In the present work the CNN model was proposed for the classification of some dynamic and postural activities using the accelerometer and gyroscope data collected from a smartphone mounted on the waist. The current work employed three different databases, which share some characteristics including the position of the smartphone on the body, the orientation of the device, the inertial sensors and the sampling rate. Construction of the model was done layer by layer, stacking a different number of convolutional and max-pooling layers to create five different CNN architectures. This work showed a clear optimization of the CNN model by tuning the number of filters and the filter size to observe their effects on performance.

Finally, a HAR cross-corpus was performed to improve generalization. Having three databases, the CNN model with the five architectures was tested on two databases using labeled training data from the other dataset.

Chapter 3

Methods

This chapter describes the methods used to predict and identify human activities, and includes datasets description, signal pre-processing, CNN model architectures description, hyperparameters tuning and performances evaluation.

3.1 Datasets description

This section describes the datasets selected among different public databases containing kinematic data of human subjects. The selected datasets are the most common used in human activity research fields based on wearable sensors [26], [11], [30] and share some characteristics, including the position of the smartphone on the body, the orientation of the device, the sensors and the sample rate.

3.1.1 Human Activity Recognition Using Smartphones Data Set

The Human Activity Recognition Using Smartphones Data Set is one of the datasets readily available on UC Irvine (UCI HAR) Machine Learning Repository. The dataset includes inertial signals collected from 30 people aging from 19 to 48, while wearing a waist-mounted smartphone (Samsung Galaxy SII) embedding both a 3-axial accelerometer and a 3-axial gyroscope; data was captured at a steady rate of 50 Hz. Each subject performed six daily activities (ADL), namely: walking, walking upstairs, walking downstairs, sitting, standing, laying. In the present work, walking, sitting, and standing were selected from the entire activity set. All the others activities were merged together, forming the 'other' class. First, sensor signals were pre-processed with a median filter and a third order low-pass Butterworth filter, with a cutoff frequency of 20 Hz.
Then, data was splitted into fixed time windows of 2.56 s, with 50% overlap to create windows with 128 time-steps. Thus, each window includes six variables, corresponding to the 3-axis of acceleration and 3-axis of angular velocity. After signals pre-processing, several time and frequency features were extracted, leading to a total of 561 features commonly used in the field of human activity recognition. Finally, the dataset has been splitted into a training set (70%) and a test set (30%).
Figure 3.1 shows the time associated to each activity in UCI HAR database. The x-axis refers to the activities identifiers, ranging from 1 to 4, corresponding to walking, sitting, standing and other, respectively. The y-axis represents the duration in seconds of each activity performed by all subjects.

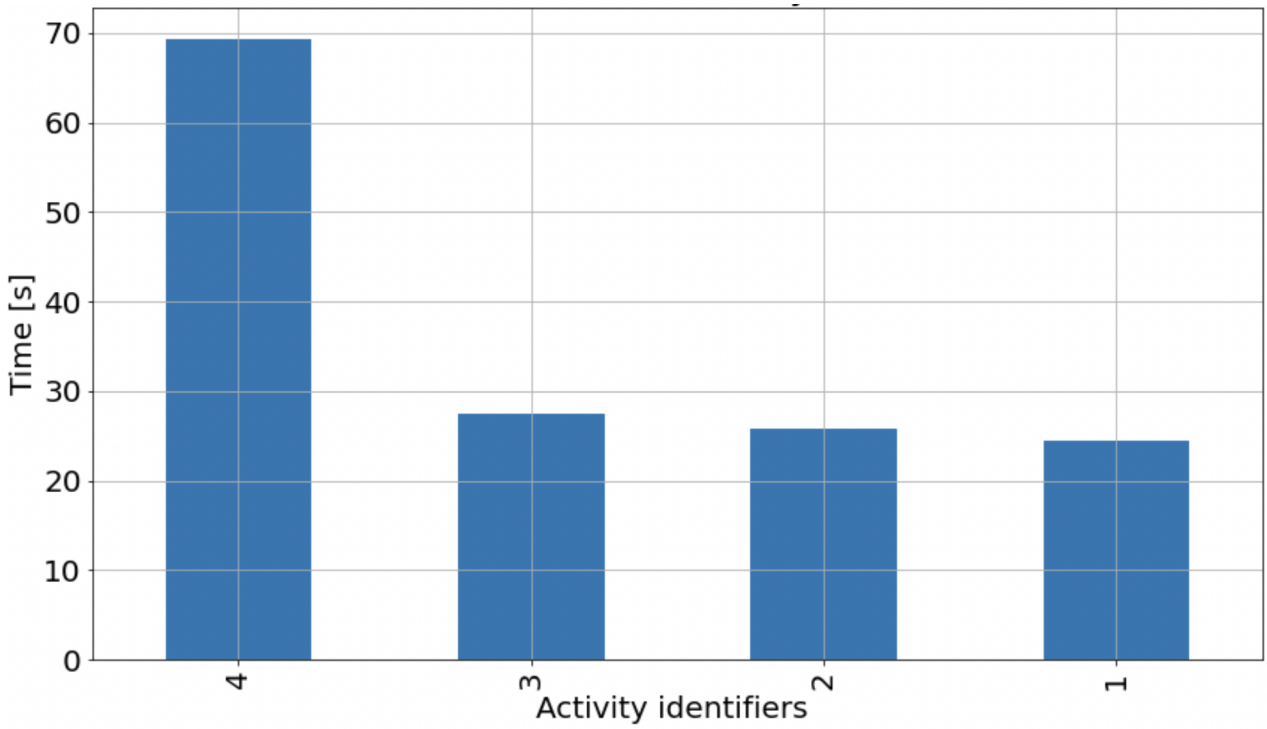

Figure 3.1: Duration associated to each activity in UCI HAR database. The 'other' class, standing, sitting and walking correspond to the identifiers 4, 3, 2, and 1, respectively.

3.1.2 Human Activity Postural Transitions Data Set

The Human Activity Postural Transitions (HAPT) dataset is an update version of the UCI Human Activity Recognition Using Smartphones dataset, that provides the original raw tri-axial signals recorded by 30 volunteers within an age bracket of 19-48 years. By using inertial sensors in a smartphone placed on the waist, 3 axial linear acceleration from an accelerometer and 3-axis angular velocity from gyroscope data in three dimension (x,y,z) were recorded at constant frequency rate of 50 Hz, without any signal pre-processing process, in order to make predictions with the raw data. The activities performed by each subject were divided into three static postures (standing, sitting, lying), three dynamic activities (walking, walking downstairs and walking upstairs) and six postural transitions (stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand). In this work, only five activities were selected from this database, while others are merged together to form the 'other' class. The selected activities included walking, sitting, standing, stand-to sit, and sit-to stand.
Figure 3.2 reports the duration in seconds of each activity in HAPT database.

3.1.3 RealWorld Human Activity Recognition Data Set

The RealWorld Human Activity Recognition (RealWorld HAR) dataset includes sensor data from a 3-axial accelerometer and 3-axial gyroscope, sampled at a fixed frequency rate of 50 Hz. The recorded activities include climbing stairs down and up, jumping, lying, standing, sitting, running, and walking, performed by 15 subjects aging from 16 to 62 years. In this work, walking, sitting, and standing were selected, while the remaining activities were merged together to form the 'other' class. The database provides data recordings from 7 different sensors on the body, including chest, head, upper-arm. In the present study, only data from the sensor on the waist was selected.

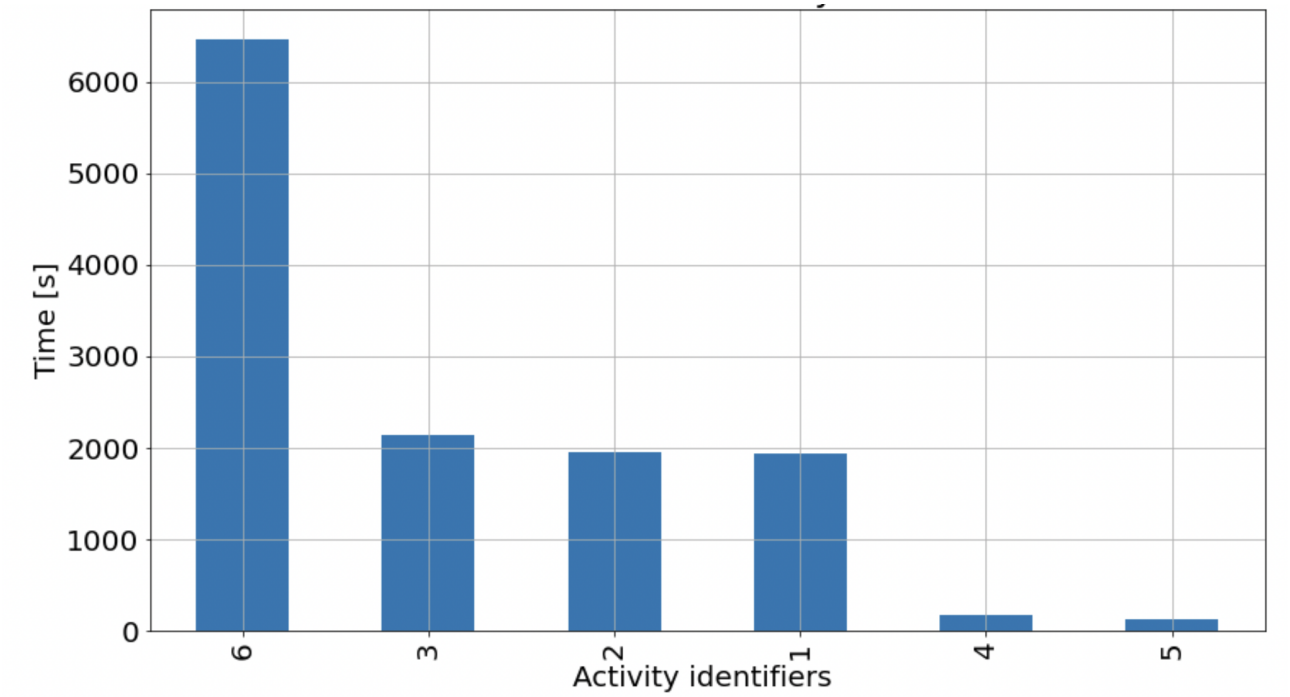

Figure 3.2: Duration associated to each activity in HAPT dataset. Walking, sitting, standing, stand-to-sit, sit-to-stand, and the 'other' class correspond to the identifiers ranging from 1 to 6.

Figure 3.3 shows the duration in seconds (s) of each activity, performed by all subjects in RealWorld HAR dataset.

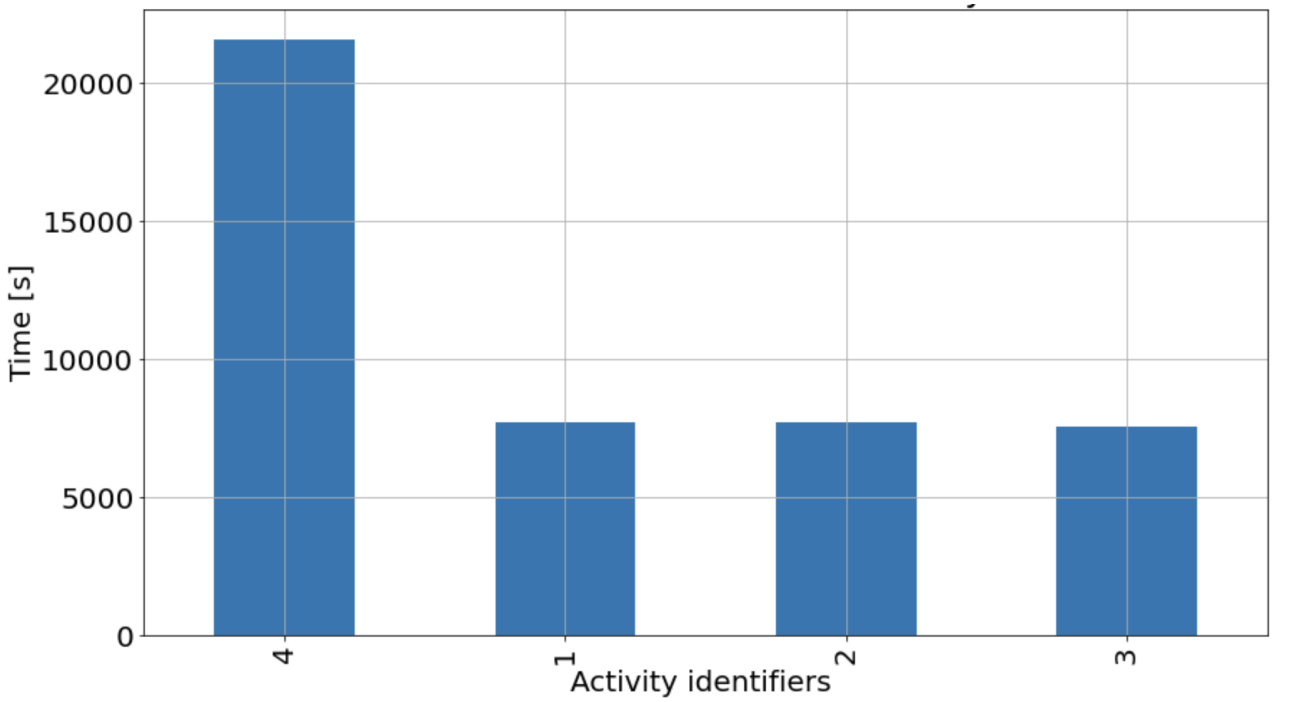

Figure 3.3: Duration associated to each activity in RealWorld HAR dataset. Walking, sitting, standing, and the 'other' class correspond to the identifiers 1,2,3 and 4.

Table 3.1 summarizes all features of each database and itemizes the number of subjects, their age, the sensor type and location on the body, the sample rate, and finally the activities selected in the present work.

Dataset	Subjects	Age (years)	Sensor type	Sensor location	Sample rate (Hz)	Selected activities
UCI HAR	30	19-48	accelerometer and gyroscope	on waist	50	walking, sitting, standing
HAPT HAR	30	19-48	accelerometer and gyroscope	on waist	50	walking, sitting, standing, stand to sit, sit to stand
RealWorld HAR	15	16-62	accelerometer and gyroscope	on waist	50	walking, sitting, standing

Table 3.1: Description of the three datasets used in this work.

3.2 Data pre-processing

First of all, the data structure of the three dataset was uniformed, together with the class label indicators.

3.2.1 Data structure

In each database, sensor data was collected from a 3-axial accelerometer and 3-axial gyroscope. The accelerometer detects the acceleration force applied to the sensor along the three physical axes (x,y,z), and it is measured in m/s^2. On the other hand, the gyroscope provides orientation details and measures the sensor rotation speed around the three physical axes (x,y,z). The angular velocity is measured in rad/s.
The following data structure was set and used for all the datasets. The 3-axial accelerometer readings were located in the first three of columns of a matrix, followed by the 3-axial gyroscope readings; the final column referred to the label activity. **Table 3.2** shows an example of the data structure of each database.

Acc x-axis	Acc y-axis	Acc z-axis	Gyro x-axis	Gyro y-axis	Gyro z-axis	label

Table 3.2: Data structure of each database.

Once identified the proper data structure, data from each database was splitted into a training set (80% of the total data) and a test set (20% of the total data). Pre-processing was performed equally on noth the training and the test set.

Figure 3.4 shows 3-axial accelerometer signals of 'walking' of the HAPT database for the first 500 samples, that correspond to 10 s of duration.

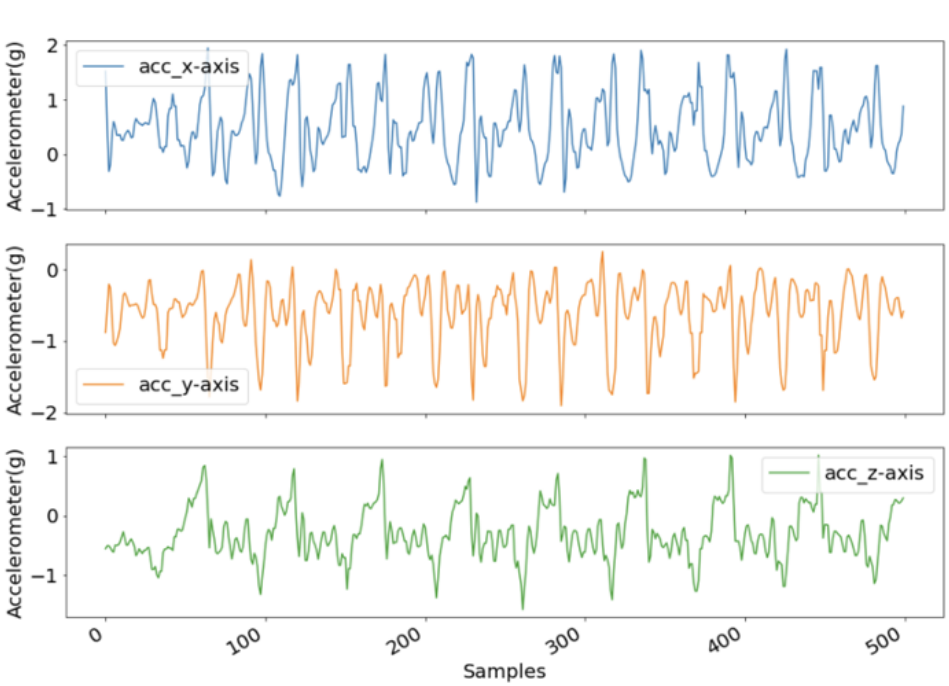

Figure 3.4: 3-axial accelerometer signals of 'walking' for the first 500 samples of HAPT database.

Figure 3.5 shows 3-axial gyroscope signals of 'sitting' of the HAPT database for the first 500 samples, that correspond to 10 s of duration.

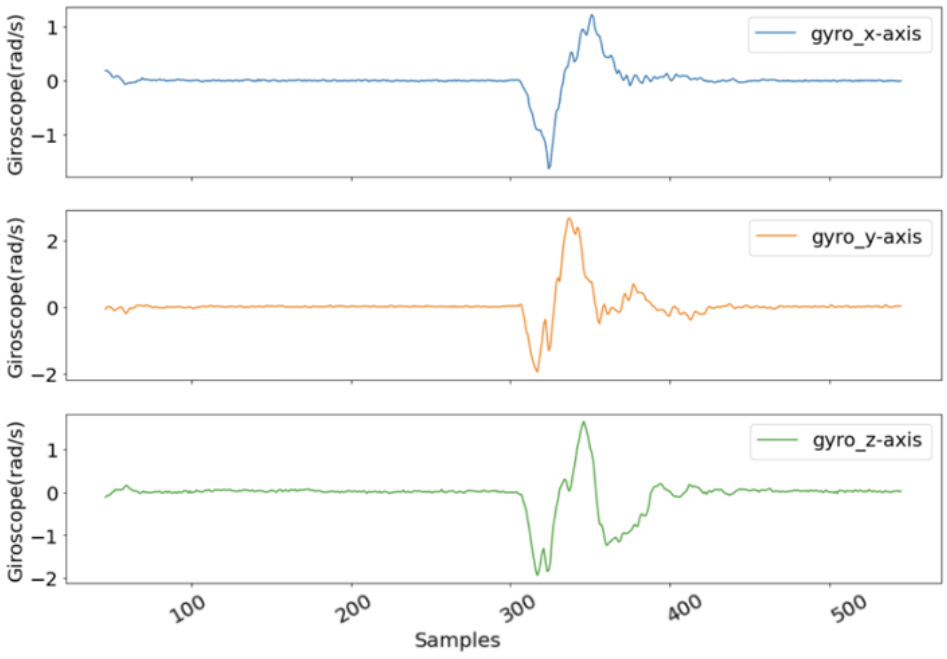

Figure 3.5: 3-axial gyroscope signals of 'sitting' for the first 500 samples of HAPT database.

3.2.2 Data standardization

Scaling is a common requirement for Deep Learning Neural Networks to improve model stability and performances.
Unscaled input variables could lead a slow or unstable learning process, whereas unscaled target variables could cause the learning process to fail.
The scaling used in the present data is z-score standardization, which maps the data into a distribution with mean 0 and a standard deviation 1, as reported in Equation 3.1.

$$x' = \frac{x - \mu_x}{\sigma_x} \tag{3.1}$$

where $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean value and $\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$ is the standard deviation.

3.2.3 Data segmentation

Segmentation is the most common pre-processing method implemented for sensor-based HAR and plays a crucial role in improving the performances of HAR applications. The segmentation approach used in the current work is based on ***Sliding Windows***, which are consecutive and equal segments of fixed size so that each of them is analyzed separately and sequentially. Each segment includes sufficient characteristics that allow the recognition of human activity at a given moment. The windows' size was set to 2.56 s, which was found to provide good performances on the training process, considering a frequency rate of 50 Hz.
In addition windows were overlapped with a percentage of 50%, in order to create segments in which 50% of samples of the previous window intersects the samples of the next window.
Figure 3.6 depicts the *Sliding Window* technique, in which the dataset is defined as $X = D_1, D_2, ..., D_t$, where D_t represents the data value at time t. Each window is defined as W_i with a Sliding Window size w of 2.56 s, while the overlap is set on 50%.

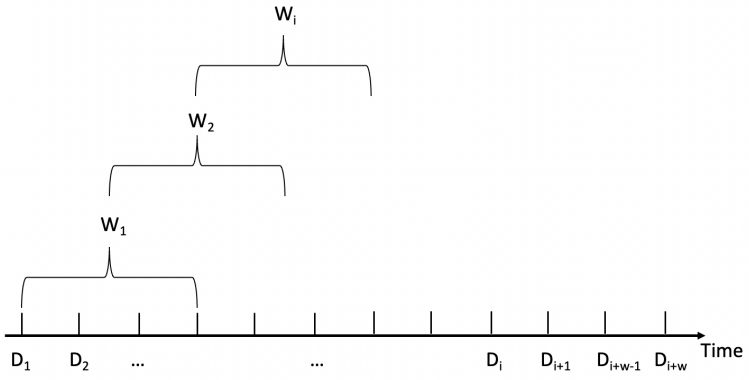

Figure 3.6: Sliding Windowing technique. Each time window W_i with a window size w of 2.56 s slides forward for a portion of data, in which D_t represents the data value at time t.The overlapping segment was set on 50%.

After data segmentation, the ***Hold out*** technique was applied on the training set, which consists in dividing the training set into the train set (80% of the training set) and the validation set (20% of the training set).
The *Hold out* technique is a simple technique to implement for the performance evaluation of machine learning models, and works very well on large datasets. **Figure 3.7** visually describes such a method. **Figure 3.8** summarizes all signals pre-processing steps performed on each

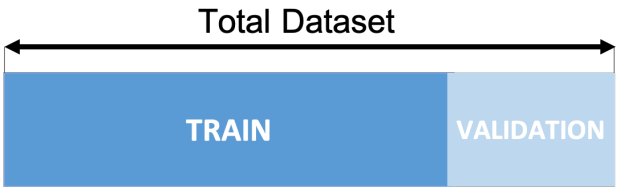

Figure 3.7: Hold out technique: the training set of the total dataset is splitted into two parts: 80% as training set and the remaining 20% as validation set.

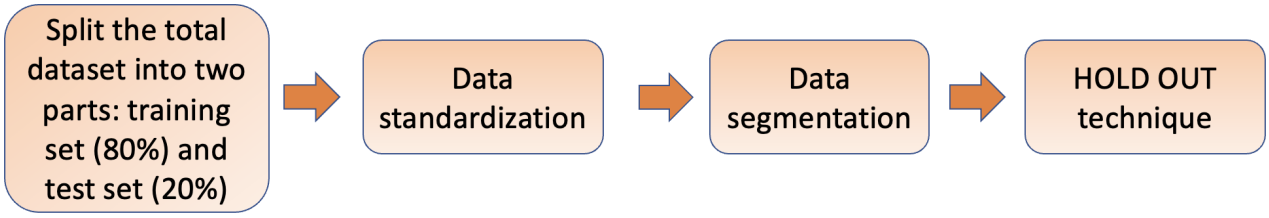

Figure 3.8: Signals pre-processing steps.

database. In order to ovoid over-fitting on RealWorld HAR Dataset, ***Leave-One-Subject-Out*** **(LOSO)** validation was performed. LOSO is an algorithm that iteratively trains the model with data from all subjects except one, which is used for validation. In the present work, LOSO was performed both for validation and for testing. Specifically, LOSO is used first for the model optimization, i.e. architecture determination and hyperparameters tuning, and then for testing and for the final performance evaluation. **Figure 3.9** visually describes LOSO.

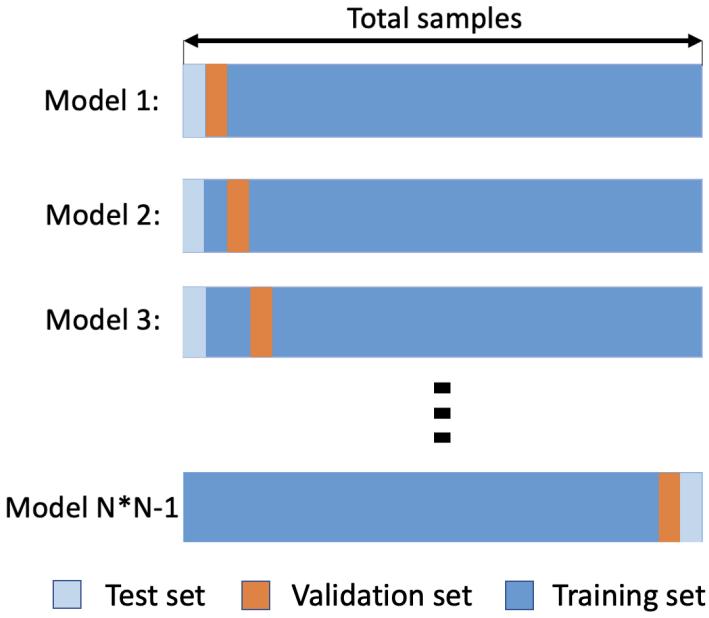

Figure 3.9: Leave-One-Subject-Out technique (LOSO). Iteratively some samples are selected as train set and the remaining as validation and test set.

Figure 3.10 summarizes the entire process of signals pre-processing for the RealWorld HAR database.

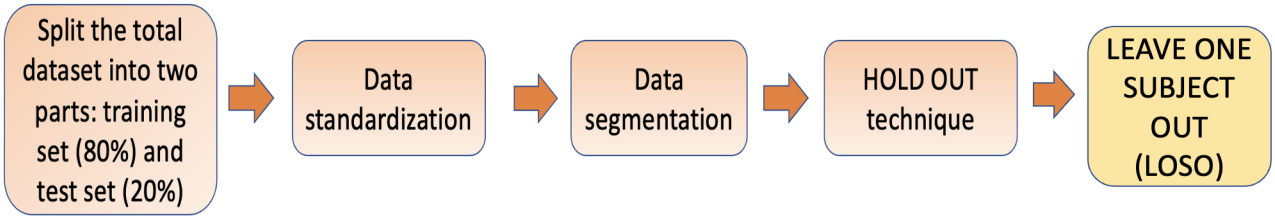

Figure 3.10: Signals pre-processing steps for the RealWorld HAR database.

3.3 Convolutional Neural Network

In the present work, Convolutional Neural Network (CNN) was selected among Deep Learning network models.
CNN learn directly from data, eliminating the need to manually extract features. Similarly to other neural networks, CNNs include an input layer, an output layer, and many hidden intermediate layers. In addition, differently from other neural networks, CNNs have a number of **convolutional layers**, where filters slide along the input data. The convolutional layer is the key component of CNNs and is generally the first layer of the architecture. Such layer performs the convolution operation, in which a filter slides along the input data to generate an output value for each position input, by executing the scalar product between the filter and the input. The filters corresponds exactly to the extracted features. If $\mathbf{x}$ is input data and $\mathbf{f}$ is the filter, the convolution operation's result is reported in Equation 3.2.

$$\mathbf{c} = f \cdot x \tag{3.2}$$

Generally, a convolutional layer is made of a set of N_F filters, each of which convolves along the width and height of the input volume. Therefore N_F two-dimensional activation maps are produced, each providing the result of the relative filter in each spatial position. Their concatenation along the third dimension produces the output of the convolutional layer. The number N_F of filters, that compose the convolutional layer is one of the hyper-parameters that can be tuned to obtain high model performances. On the other hand, the stride or kernel size is the hyper-parameter that specifies how many input time is processed into the feature maps. When the stride is equal to one, the filter slides one pixel at a time, and consequently a longer output is generated. On the contrary, higher values of kernel size move the filter with high jumps, thus a smaller output is generated.
The convolution layer is generally alternated by a non linear activation function. The most common activation function used is the ***Rectified Linear Unit (ReLU)***, defined as

$$ReLU(net) = max(0, net) \tag{3.3}$$

This function was selected over other functions, such as the sigmoid or hyperbolic tangent, due to its capability to avoid network saturation. This allows to prevent inputs with different characteristics from causing the same type of output.
Formulae of *sigmoid* function is reported in Equation 3.4:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{3.4}$$

The other layers that compose CNNs are described below.

The **pooling layer** performs an aggregation of information in the input volume, generating feature maps of inferior size. The main task of "pooling" is reducing the number of parameters by consolidating them only to the most essential elements. In addition, it reduces the computational load, speeds up the calculation and makes some of the features detected more robust. Widespread pooling techniques include average pooling and max pooling, which compute respectively the average and the maximum value of data. The max-pooling layer was selected in the current work, being the most used in the applications.

The **dropout layer** is a regularization technique, that randomly ignores or 'drops out' some hidden neurons during the training process [8]. This is a simple way to prevent neural networks from *over-fitting* [9], thus avoiding loss of performance on the test data. If the problem of over-fitting occurs, the model learns patterns that are more specific for a given dataset, thus losing the generalization capability. Applying dropout, the learning process slows down and the model achieves better performances on completely new unknown data.

The **batch normalization layer** is meant to accelerate the computation, provides regularization, and reduces the generalization error. Through the use of batch normalization layer the network model achieves stable training performance [10].

The **fully connected layer** connects every neuron in one layer to every neuron of the next layer. The output of this layer is flattened into a one-dimensional vector and used for the classification [11].

The **softmax layer** computes the probability distribution on predicted classes. The model makes the prediction based on the output classes probability. The model output is defined as a vector of elements containing the probability of a given window belonging to each type of activity.

3.4 Convolutional Neural Networks architectures

This section describes the CNN architectures proposed in the present work. CNN was developed on five different architectures having different layers combinations. The different architectures are listed and described below, sorted in ascending order of complexity.

1. The first architecture of the CNN model consists of *1D CNN* layer, followed by *ReLU activation function* and a *max-pooling* layer, that reduces the learned features to half their size. Then, the learned features pass through a *dropout* layer for regularization and a *batch normalization* layer to accelerate training process. Finally, the learned features are flattened to one long vector through the *flatten* layer, that serves as a connection between previous layers and dense layers. *Dense* is the layer used for the output layer, which makes a prediction. *Softmax* is the final layer, that returns the probability for each class.
   Figure 3.11 shows the first architecture of the CNN model.

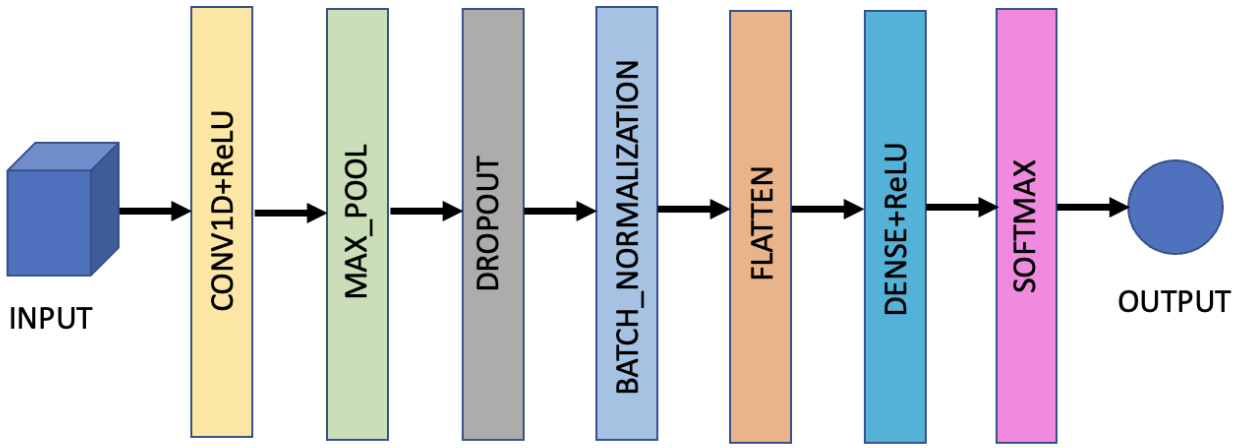

Figure 3.11: First architecture of the CNN model. The 3-dimensional input passes through the first architecture of the CNN model and output is computed as an integer for the class number.

2. The second architecture of the CNN model is composed by two identical blocks made of *1D CNN layer*, *ReLU activation function*, *max-pooling* layer; then, dropout and batch normalization layers are applied. Finally, flatten, dense and softmax are applied to get the final output. **Figure 3.12** represents the second architecture of the CNN model.

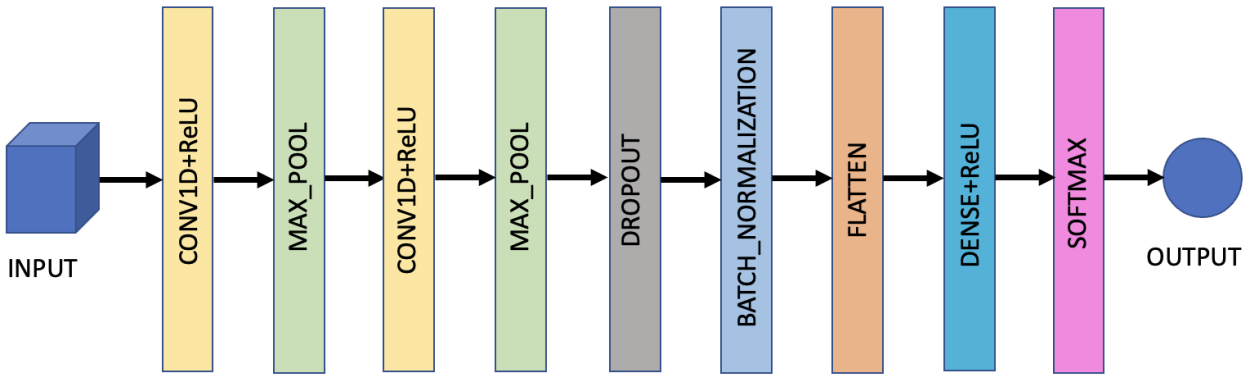

Figure 3.12: Second architecture of the CNN model. The 3-dimensional input passes through the second architecture of the CNN model and output is computed as an integer for the class number.

3. The third architecture of the CNN model consist of three identical blocks of *1D CNN* layers, *ReLU activation function*; then dropout and batch normalization layers are applied. Finally, flatten, dense and softmax are applied to get the final output. **Figure 3.13** shows the third architecture of the CNN model.

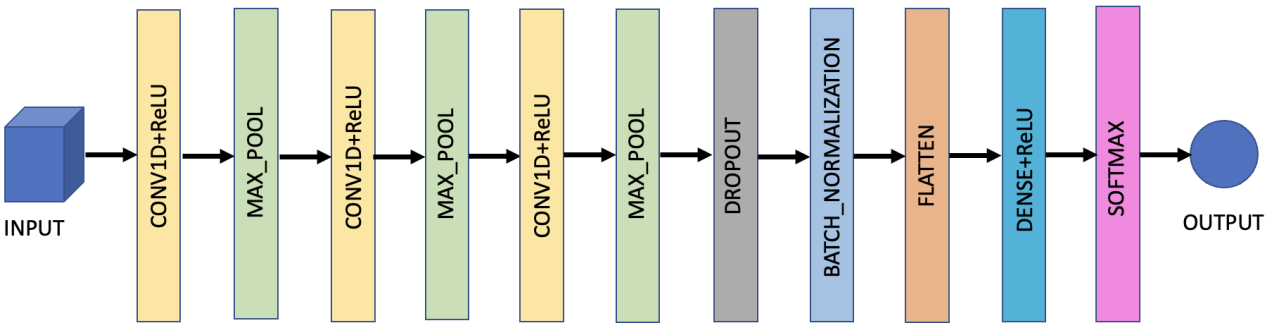

Figure 3.13: Third architecture of the CNN model. The 3-dimensional input passes through the third architecture of the CNN model and output is computed as an integer for the class number.

4. The fourth architecture of the CNN model is composed by two identical blocks of*1D CNN* layers, followed by *ReLU activation function*. Then, a max-pooling layer is applied. Subsequently, dropout layer and batch normalization are applied. Finally flatten, dense and softmax are applied to get the final output. **Figure 3.14** shows the fourth architecture of the CNN model.

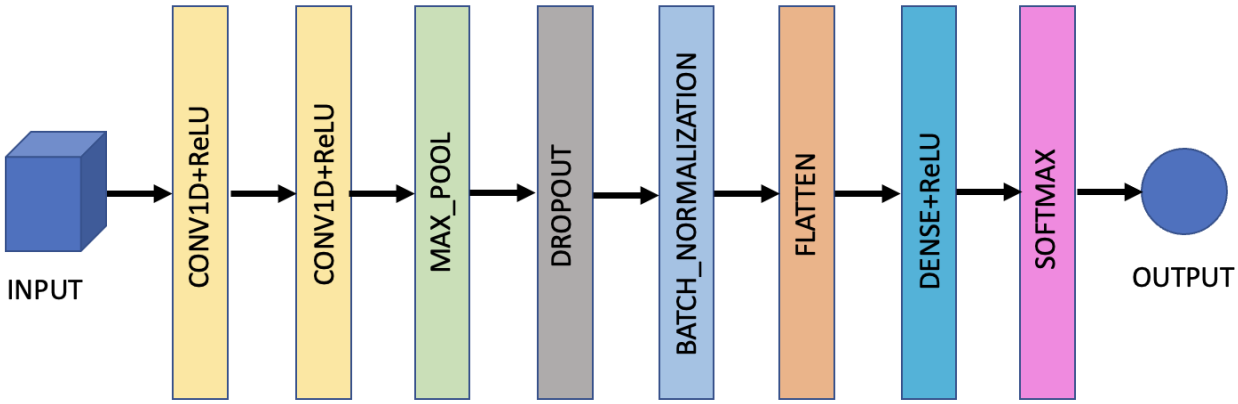

Figure 3.14: Fourth architecture of the CNN model. The 3-dimensional input passes through the fourth architecture of the CNN model and output is computed as an integer for the class number.

5. The fifth architecture of the CNN model consists of two *1D CNN* layers, followed by a *ReLU activation function*. Then a max-pooling layer and a *1D CNN* layer, followed by a *ReLU activation function* are applied. Subsequently, dropout layer and batch normalization are applied. Finally flatten, dense and softmax are applied to get the final output. **Figure 3.15** shows the fifth architecture of the CNN model.

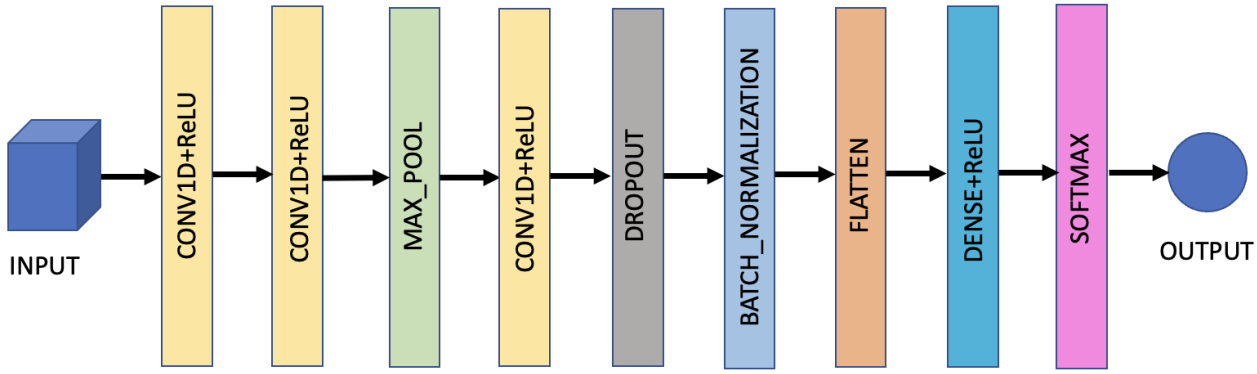

Figure 3.15: Fifth architecture of the CNN model. The 3-dimensional input passes through the fifth architecture of the CNN model and output is computed as an integer for the class number.

3.5 Training and Validation

In the following, the optimizer, the loss function, and the optimization metric are listed and described in details.

- **Adam** is a common used optimizer that adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A small learning rate may lead to more accurate weights, but the time it takes to compute the weights will be longer. Moreover, such method has various advantages [7], including
  - simple implementation
  - computationally efficiency
  - little memory requirements
  - suitability for problems that are large in terms of data and/or parameters,
  - fast tuning
- **Categorical cross-entropy** was selected as loss function, computed as reported in Equation 3.5.

$$Loss = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i \tag{3.5}$$

  where $\hat{y}_i$ is i^th scalar value in the model output, y_i is the corresponding target value, and the output size is the number of scalar values in the model output.
  Categorical cross-entropy is well suited to classification tasks.

- **Accuracy** was the metric selected to evaluate the model after the training process. It calculates how often predictions on validation set are equal to labels. *Accuracy* is defined as the percentage of correct predictions on the validation set and can be calculated easily by dividing the number of correct predictions on the validation set by the number of total predictions.

$$Accuracy = \frac{\text{correct predictions on validation}}{\text{total predictions}} \tag{3.6}$$

Finally, the CNN model was trained on the training set, optimizing the hyper-parameters, and then evaluated on the validation set.

3.6 Hyper-parameters tuning

To generate high performances and fast output, the CNN model needs to be optimized performing hyper-parameters tuning. The hyper-parameters should be chosen carefully, because they could negatively affect the performances of the model. In the following, parameters of the model architecture are listed and described.

- **Number of layers** is the first hyper-parameter to tune in a Convolutional Neural Network. A larger number of layers makes the network more complex and depth, due to the fact that layers learn progressively more complex and specific features. The number of layers was tuned from 1 to 3, combining convolutional layers and max-pooling layers.
- **Number of filter maps** represents the number of times the input is processed. The number of filter maps used in this work assumed the values reported below.

$$\text{features maps} = [8, 16, 32, 64, 128, 256] \tag{3.7}$$

- **Size of the kernel** consists of the size of the filter which convolves around the feature maps. The list of values used for kernel size is as follows.

$$kernel = [2, 3, 5, 7, 11] \tag{3.8}$$

- **Stride and pooling size** are hyper-parameters of the pooling layer and are kept steady at 1 and 2, respectively.
- **Dropout rate** is the hyper-parameter of the dropout layer. It was kept constant at a value equal to 0.5, that is the common value used in CNNs [8] [9].
- **Batch size** is the hyper-parameter referred to the number of training examples used at each iteration. It could affect the learning algorithm, especially about the speed which a model learns and the stability of the learning process. A *batch size* of 32 was chosen, as works well generally [10].
- **Number of epochs** is set to 10, that is the best compromise between generalization and accuracy both on the training and the test set.

3.7 Prediction

The model trained on the training set and evaluated on the validation set was used to make predictions on the test set and identify the class for this new data. Class prediction was performed not only on test set of each database, but also on the other two entire datasets. Classes predictions on the two databases allow to see how well the model works and generalizes on new data.

Table 3.3 lists the values of selected hyper-parameters.

Stage		Hyper-parameters	Selected Values
Data pre-processing	window size		2,56s
	overlap		50%
CNN model	*Convolutional layer*	number of filters maps	8,16,32,64,128,256
		kernel size	2,3,5,7,11
	Max-pooling layer	stride	1
		pooling size	2
	Dropout layer	dropout rate	0.5
	Batch normalization	batch size	32
Training and Validation	optmizer		Adam
	loss function		Categorical cross-entropy
	metric		Accuracy
		number of epochs	10

Table 3.3: List of selected hyper-parameters.

Figure 3.16 summarizes the algorithm approach used to predict classes on the three different databases.

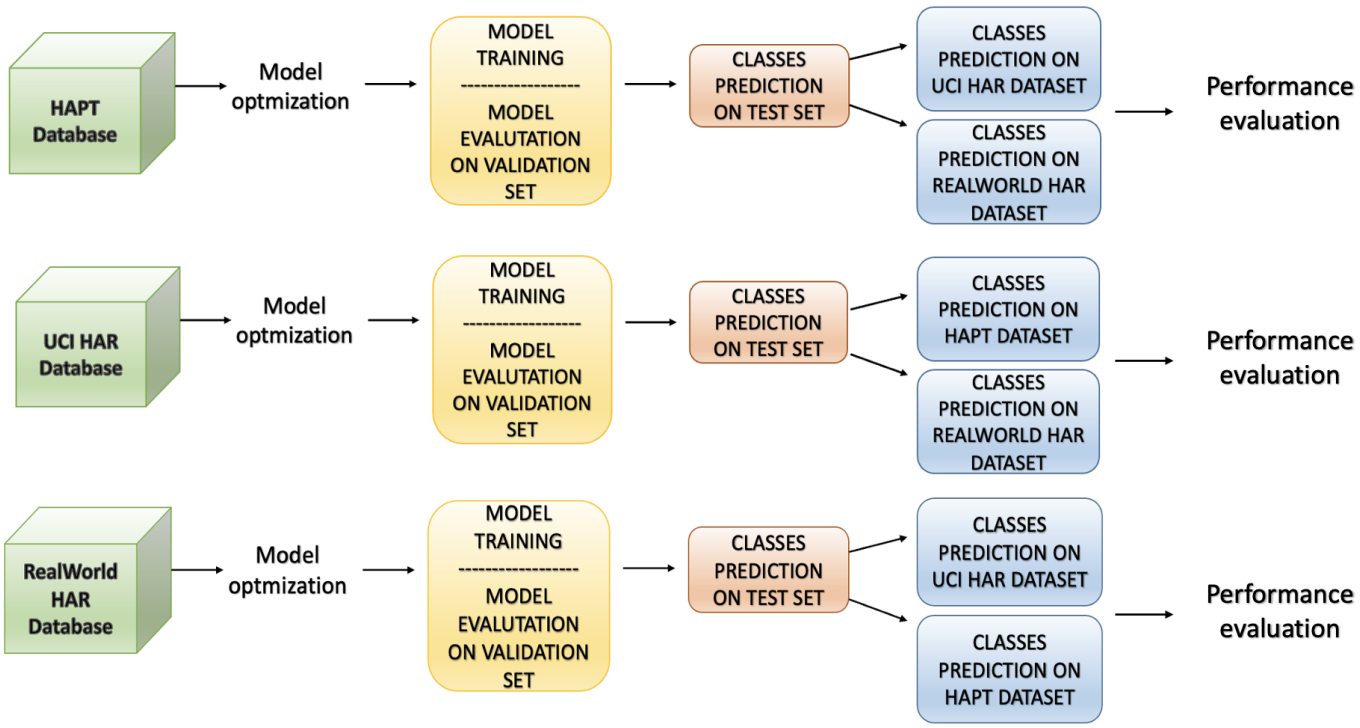

Figure 3.16: Algorithm approach.

3.8 Performance Evaluation

In order to provide an exhaustive performance evaluation and to compare results from the various architectures of each dataset, different performance metrics were computed.
First of all, the **Confusion Matrix**, also known as an error matrix, is often used to visualize the performance of a classification model and determines how accurate and effective the model is. The matrix a number of rows corresponding to the actual classes and a number of columns representing the predicted classes. To better understand the connections between actual and predicted values, the following terms are associated with the confusion matrix.

- True positives (TP): model correctly predicts the positive class (prediction and actual both are positive)
- False positives (FP): model correctly predicts the negative class (prediction and actual both are negative).
- True negatives (TN): model gives the wrong prediction of the negative class (prediction positive, actual negative)
- False negatives(FN): model wrongly predicts the positive class (prediction negative, actual positive).

The following metrics were computed for performance evaluation, and reported as percentages in Chapter 4. Metrics value ranges from 0 to 1, with 1 representing the best result.

- **Accuracy** is a common evaluation metric, that is calculated as the ratio between the number of correct predictions and all predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

- **Specificity** is calculated as the number of negative predictions divided by the total number of negatives.

$$specificity = \frac{TN}{TN + FP} \tag{3.10}$$

- **Precision** is the ability of a classifier not to label a positive instance that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

$$precision = \frac{TP}{TP + FP} \tag{3.11}$$

- **Sensitivity (or Recall)** is the ability of a classifier to correctly identify all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.12}$$

  Sensitivity is usually used to limit the number of false negatives.

- f_1**-score** is a weighted harmonic mean of the precision and recall metrics, and it also ranges from 0 to 1.

$$f_1\text{-score} = 2 \cdot \frac{\text{sensitivity} \cdot \text{precision}}{\text{sensitivity} + \text{precision}} \tag{3.13}$$

3.9 Selection of the best architecture

After computing performances for each architecture of each database, the best architecture for each architecture was selected in terms of maximum validation accuracy. Between architectures with the same validation accuracy, it was selected that with the lower computational complexity.

LOSO process is reported in **Algorithm 1**.

Algorithm 1 Algorithm for model optimization, validation and test performance evaluation

procedure OPTIMIZEDMODEL($Data$),PERFORMANCE($Data$) ▷

  for $i \leftarrow 1$ to N **do** ▷ Perform N times test procedure

    for $j \leftarrow 1$ to $N-1$ **do** ▷ Perform N-1 times validation procedure

      $[trainingSet] \leftarrow$data from all subject except for j^{th} ▷

      $[validationSet] \leftarrow$ data from j^{th} subject ▷

      for $filtersfunction \leftarrow$ [8,16,32,64,128,256] **do** ▷ tune filter maps

        for $kernelfunction \leftarrow$ [2,3,5,7,11] **do** ▷ tune kernel size

          $Model \leftarrow train(model(trainingSet))$ ▷ train model

          $prediction \leftarrow predict(model(validationSet))$ ▷ predict validationSet

        end for

      end for

      $ACC \leftarrow \frac{\text{correct predictions on validationSet}}{\text{total predictions}}$ ▷ compute ACC

      $[filtersfunction, kernelfunction] \leftarrow max(ACC)$ ▷ optimal parameters

      $optimizedModel \leftarrow model(filtersfunction, kernelfunction)$

    end for

    $performance \leftarrow predict(model(i^{th}subject))$ ▷ compute performance on testSet

  end for

  return $optimizedModel, performance$ ▷

end procedure

Experiments were performed on a PC with macOS 11 Big Sur operating systems, with 8GB RAM and GPU 8-core. Matlab version R2021a was used to create data structure of each database. Google Colab was used for CNN training and testing, with Python version 3.7.12, Keras packages with TensorFlow backend to the implementation of the deep learning model.

Chapter 4

Results

This chapter discusses the effect of the number of layers, feature maps and kernel size. Additionally, it presents and analyzes the results obtained from the training process and the model evaluation for each architecture of each database. Each architecture is described with its acronyms, where C indicates the Convolutional layer and P refers to Max-pooling layer.

4.1 The effect of the number of layers

The number of layers of the CNN model and their combination can affect the execution time and the model accuracy, as reported in the **Figure 4.1**, where the filter size and the kernel size were selected conventionally as 32 and 5, respectively.

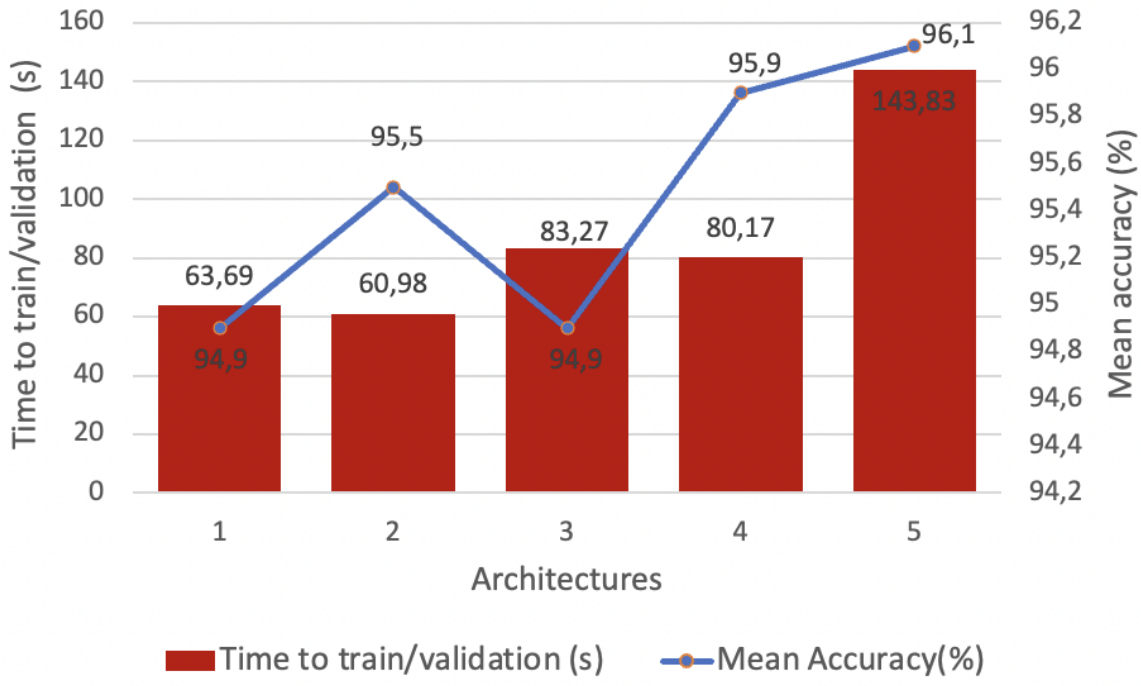

Figure 4.1: The effect of the number of layers of the CNN model.

Figure 4.1 suggests an increasing of the number of the layers of the CNN model contributes to create a deeper network. From the **Figure 4.1**, it can be seen model accuracy increases going from the 1^{st} architecture to the 5^{th} architecture, except for the 3^{nd} architecture, where the model accuracy achieves the lowest value. Accuracy, in fact, starts with a value equal to 94.9% at first architecture and reaches 96.1% at the fifth architecture. However, more depth is the

network and more larger is the complexity and execution time of the network. For this reason the 3^{rd} and the 5^{th} architecture being composed by three layers have the highest computational time.

4.2 The effect of the number of feature maps

The number of feature maps can affect the total number of parameters extracted by the model and the model accuracy, as reported in the **Figure 4.2**, that shows results of the first architecture of HAPT dataset, conventionally. More higher is the number of filters and more complex

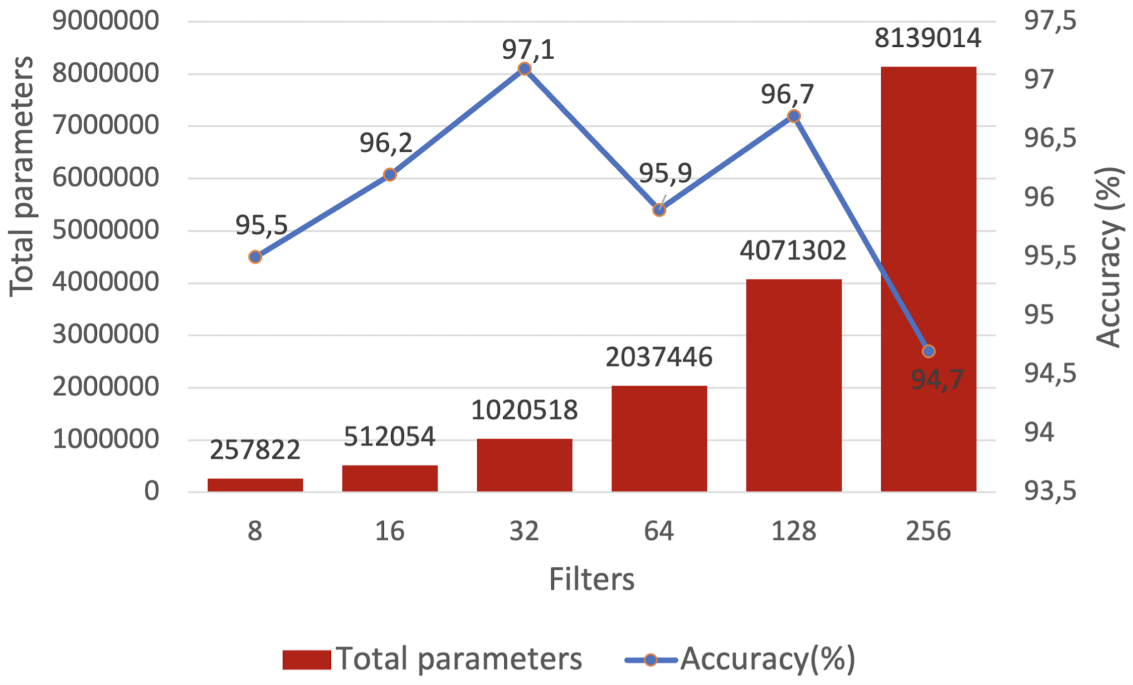

Figure 4.2: The effect of the number of filters in the first architecture of the CNN model.

and deeper is the CNN model. From the **Figure 4.2** it can be observed the increasing of the number of parameters extracted by the model with increasing the number of filter maps. From n=8 to n=256 filters, the parameters increase from 257822 to 8139014. The figure suggests the accuracy increases until n=32 feature maps, but at 64 feature maps a significant drop occurs. So, choosing of the feature maps is a critical issue.

4.3 The effect of the kernel size

The kernel size can affect the time to train and evaluate the CNN model and the model accuracy, as reported in the **Figure 4.3**. As can be observed from the **Figure 4.3** the mean accuracy increases with increasing the kernel size until k=7. Then, the model accuracy deteriorates with larger kernel size. On the other hand, the execution time increases until k=3 and then begins to decreases until k=11, where the time to train and evaluate the CNN model is just about 61 s.

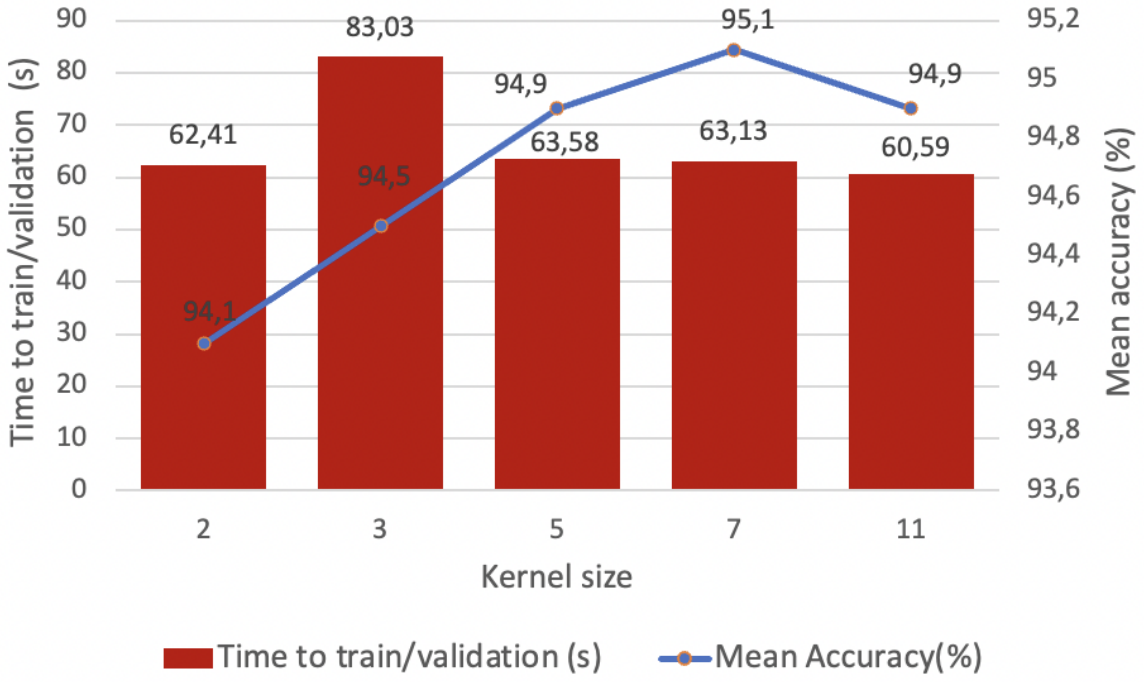

Figure 4.3: The effect of the number of kernel in the first architecture of the CNN model.

4.4 Results for each database

1. *HAPT database's results*
   Table 4.1 lists the mean accuracy in percentage on train, validation and test set for each architecture of the CNN model. **Table 4.1** suggests the best architecture in HAPT

HAPT database			
	mean accuracy (%)		
architecture	**train**	**validation**	**test**
1C+ReLU-1P	94.9	94.7	93.7
1C+ReLU-1P-1C+RelU-1P	94.9	96.0	96.9
1C+ReLU-1P-1C+RelU-1P-1C+RelU-1P	94.9	96.4	95.7
2C+ReLU-1P	96.0	96.4	94.3
2C+ReLU-1P-1C	94.5	95.4	93.8

Table 4.1: Mean accuracy(%) on train,validation and test set for each architecture of the CNN model in the HAPT database.

database may be the 3^{rd} architecture or the 4^{th} architecture, where the mean validation accuracy is the same and achieves the highest value of 96.4%. However, the 4^{th} architecture extracts a larger number of parameters (2049030 parameters) than the 3^{rd} architecture, where the number of parameters is just 440134. The 3^{rd} architecture was selected as best architecture in HAPT database for the low computational complexity. Results obtained from the third architecture are described with more detail below.
The tuning process of the number of feature maps, also called number of filters led to results reported in the **Figure 4.4**. As can be noticed from the **Figure 4.4** the median

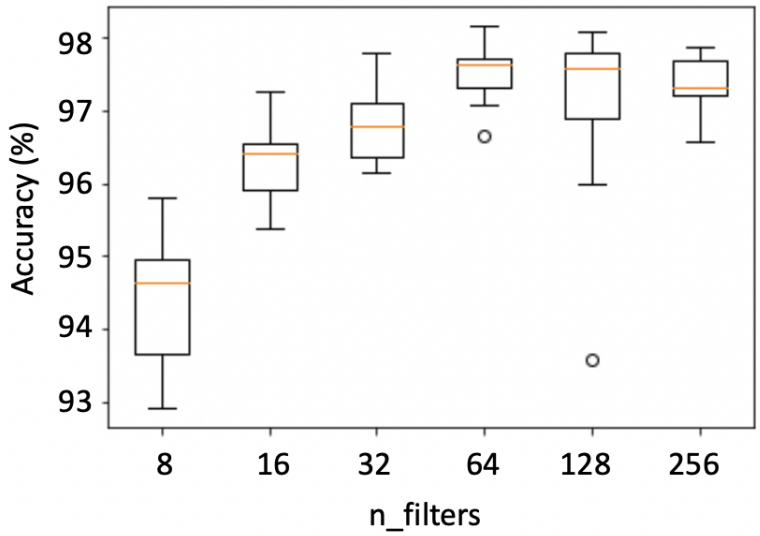

Figure 4.4: Accuracy for different number of filter maps, obtained for the third architecture in HAPT database. The best configuration, in terms of stability and computational time is given by a number of filters equal to 64.

accuracy (orange line on the box) trend gradually increases when increasing the number of feature maps, until to 64, where the accuracy reaches a peak of 97.5% with a standard deviation of ±0.40. After this peak, accuracy decreases until n=256 feature maps. A number of features equal to 64 represents the best configuration in terms of stability and computational time.
Figure 4.5 reports accuracy values for different values of the kernel size. The number of filters was set to 64, which represents the best value obtained previously.

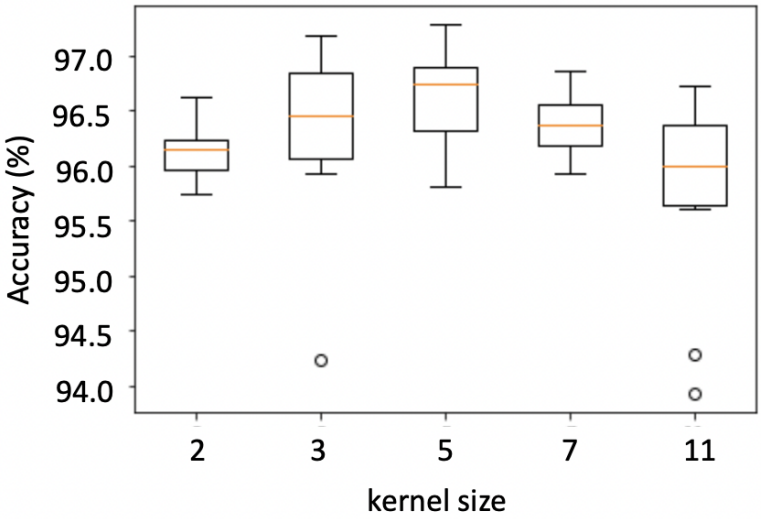

Figure 4.5: Accuracy for different size of the kernel, for the third architecture in HAPT database. Setting n = 64 feature maps, the best configuration is given by a kernel of size equal to 5.

Figure 4.5 suggests a kernel size equal to 5 is the best configuration in terms of median accuracy, that assumes a value of 96.6% with a standard deviation of ±0.46. After selecting the best configuration of feature maps and kernel size, the model was trained on the training set. **Figure 4.6** shows the different layers that compose the third architecture

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 124, 64)	1984
max_pooling1d (MaxPooling1D)	(None, 62, 64)	0
conv1d_1 (Conv1D)	(None, 58, 64)	20544
max_pooling1d_1 (MaxPooling1	(None, 29, 64)	0
conv1d_2 (Conv1D)	(None, 25, 64)	20544
max_pooling1d_2 (MaxPooling1	(None, 12, 64)	0
dropout (Dropout)	(None, 12, 64)	0
batch_normalization (BatchNo	(None, 12, 64)	256
flatten (Flatten)	(None, 768)	0
dense (Dense)	(None, 512)	393728
dense_1 (Dense)	(None, 6)	3078

Total params: 440,134

Figure 4.6: CNN model's third architecture in HAPT dataset:a number of filter set to 64 and a kernel size of 5 extract a number of parameters equal to 440134.

of the CNN model and the total number of parameters extracted. The training process was displayed in the line graph of the **Figure 4.7**, which shows the evolution of the training/validation accuracy through the epochs. As can be seen from the **Figure 4.7**, the training accuracy gradually increases while increasing the number of epochs, until reaching a peak value of about 97%, corresponding to a number of epochs equal to 10. Validation accuracy trend is very similar to that of train accuracy and achieves higher values about 98%.

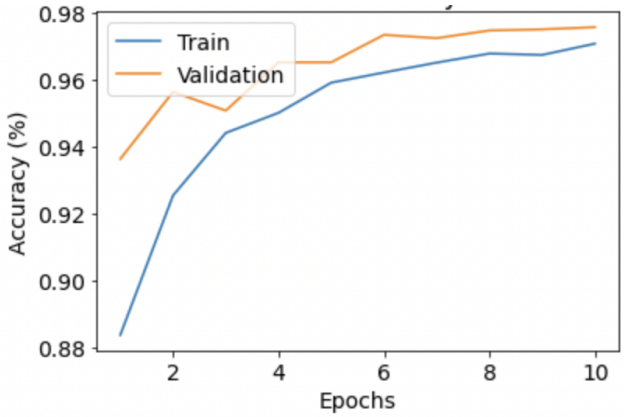

Figure 4.7: Accuracy trend on the train and validation set for the 3^{rd} architecture in HAPT dataset.

The confusion matrix computed on test set of HAPT database is reported in the **Figure 4.8**.

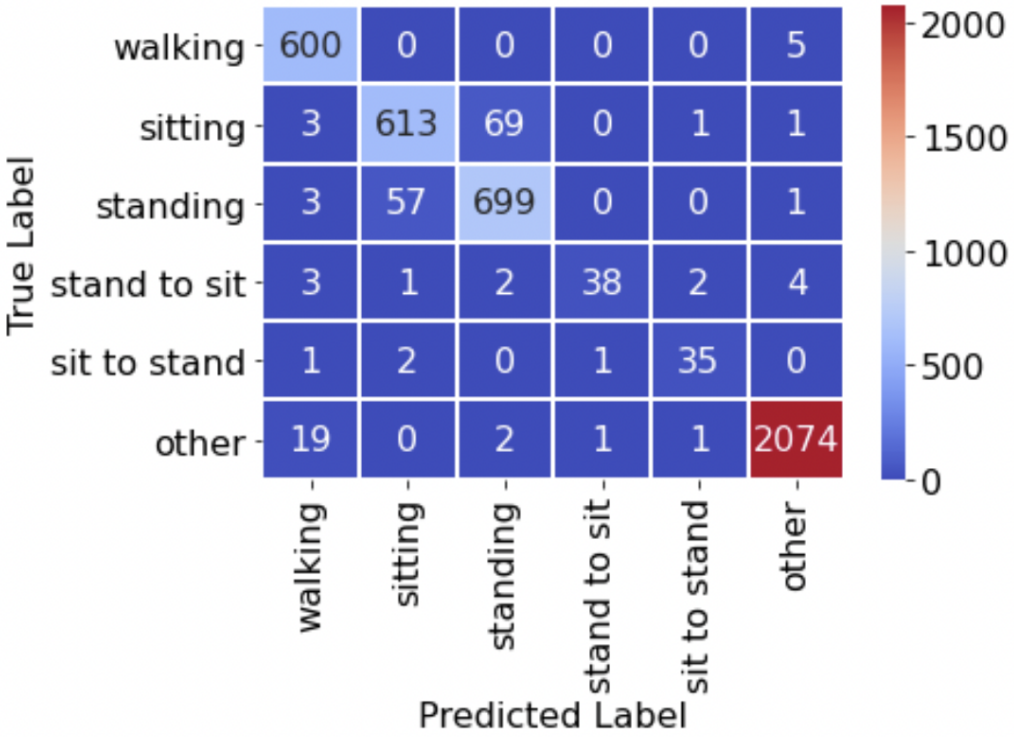

Figure 4.8: Confusion matrix computed on the test set of HAPT database in the third architecture.

Metrics computed on the confusion matrix of the **Figure 4.8** are listed in the **Table 4.2**.

activity	accuracy(%)	specificity(%)	precision(%)	sensitivity(%)	f_1-score(%)
walking	99.1	99.2	95.3	99.1	97.2
sitting	96.8	98.3	91.0	89.2	90.1
standing	96.8	97.9	90.5	91.9	91.2
stand to sit	99.6	99.9	95.0	76.0	84.4
sit to stand	99.8	99.9	89.7	89.7	89.7
other	99.1	99.4	99.4	98.9	99.1

Table 4.2: Metrics computed from the confusion matrix on the test set of HAPT database in the third architecture.

As can be observed from the **Figure 4.8** and from the **Table 4.2** the values of true positives and true negatives are very high, leading to high accuracy and specificity for all predicted classes. The values of precision, sensitivity (o recall) and f_1-score are slightly lower than those of accuracy and specificity, especially for the 'sit-to-stand' class. The wrong predictions of 'sit-to-stand' may be caused due to the fact this activity is performed for less time than other activities and this may impact the ability of the CNN model to discriminate between the activities.

From classes prediction on RealWorld HAR dataset, the confusion matrix reported in the **Figure 4.9** was obtained, in which it is evident a performances drop.

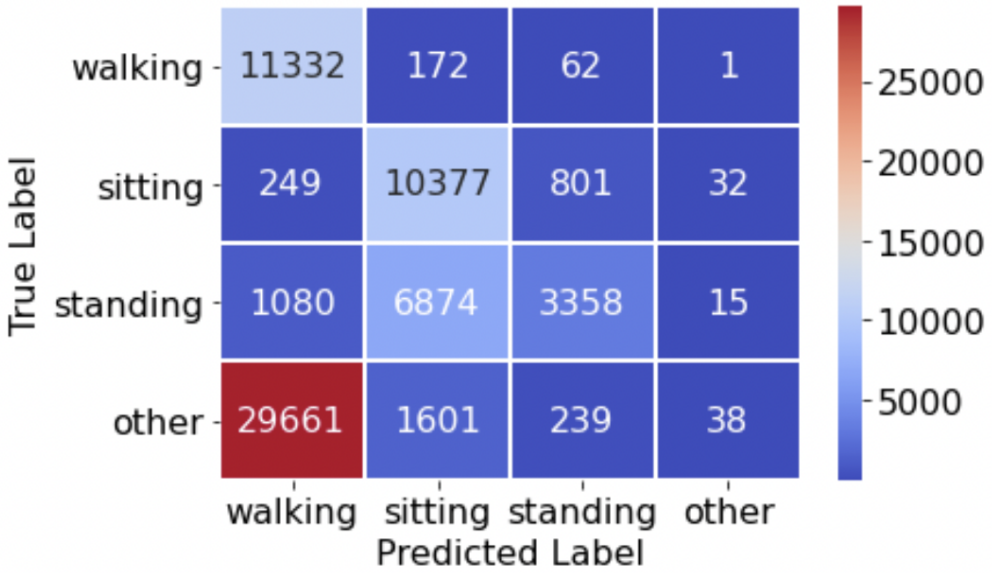

Figure 4.9: Confusion matrix obtained from classes prediction on RealWorld HAR dataset.

Metrics computed on the RealWorld HAR dataset are listed in the **Table 4.3**.

activity	accuracy(%)	specificity(%)	precision(%)	sensitivity(%)	f_1-score(%)
walking	52.6	42.9	26.7	26.7	26.7
sitting	85.2	84.1	54.5	54.5	54.5
standing	86.2	97.9	75.2	75.2	75.2
other	52.1	99.8	44.1	44.1	44.1

Table 4.3: Metrics calculated from the confusion matrix obtained from classes prediction on the RealWorld HAR dataset.

Results suggest that the model is not capable of well predicting classes on RealWorld HAR dataset, and this is evident from the low performances obtained for each class. The better performances are obtained for the 'standing' class, in which the accuracy reaches 86.2% and the specificity is 97.8%. On the other hand, precision, recall and f_1-score are about 75%. 'Sitting' class is wrongly predicted with 'standing' classes because they are similar sequence of activities and 'other' class is wrongly predicted with 'walking' class, because activities belonging to 'other' class include climbing down/up, that are similar sequence of activities with 'walking'. Summarizing, the model has mainly low performances.
Likewise, classes prediction on the UCI HAR dataset led to the confusion matrix showed in the **Figure 4.10**.
Performances obtained testing the model on the UCI HAR dataset are slightly worst than those of the RealWorld HAR Dataset, in fact precision, recall and f_1-score are just about 40%. The model wrongly predicting 'sitting' as 'standing' and vice versa. Also, 'walking' is confused with the 'other' class.

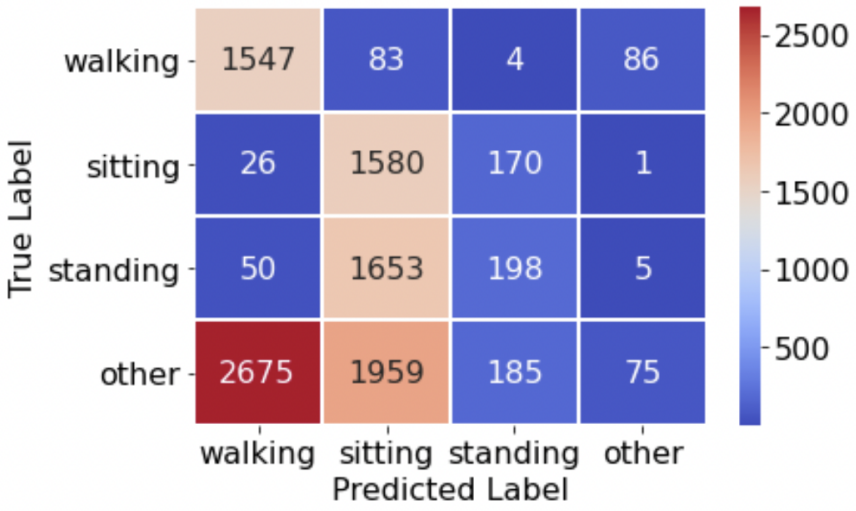

Figure 4.10: Confusion matrix obtained from classes prediction on the UCI HAR dataset.

Metrics computed on the UCI HAR dataset are listed in the **Table 4.4**.

activity	accuracy(%)	specificity(%)	precision(%)	sensitivty(%)	f_1-score(%)
walking	71.6	67.9	35.9	35.9	35.9
sitting	62.2	56.6	29.9	29.9	29.9
standing	79.9	95.7	35.5	35.5	35.5
other	52.3	98.2	44.9	44.9	44.9

Table 4.4: Metrics calculated from the confusion matrix obtained from classes prediction on UCI HAR database.

2. *UCI HAR database's results*
Mean accuracy in percentage on train, validation and test set for each architecture of the CNN model in UCI HAR database are described in the **Table 4.5**.

UCI HAR database			
	mean accuracy (%)		
architecture	**train**	**validation**	**test**
1C+ReLU-1P	72.4	73.2	71.2
1C+ReLU-1P-1C+RelU-1P	77.9	79.9	78.6
1C+ReLU-1P-1C+RelU-1P-1C+RelU-1P	75.0	77.9	77.7
2C+ReLU-1P	75.6	76.3	80.1
2C+ReLU-1P-1C	79.0	78.6	79.5

Table 4.5: Mean accuracy(%) on train,validation and test set for each architecture of the CNN model in UCI HAR database.

Table 4.5 suggests the best architecture of the CNN model in the UCI HAR database is the 2^{nd} architecture, which led to the highest mean validation accuracy of 79.9%.

Results obtained from the tuning process of the number of feature maps of the second architecture are showed in the box-plot reported in the **Figure 4.11**.
Results suggest that n = 128 may be the optimal configuration, but the computational time is very large. For this reason, the best configuration in terms of stability and computational time is 64, where the median accuracy assumes a value of of 85.6% with a standard deviation of ±1.5.

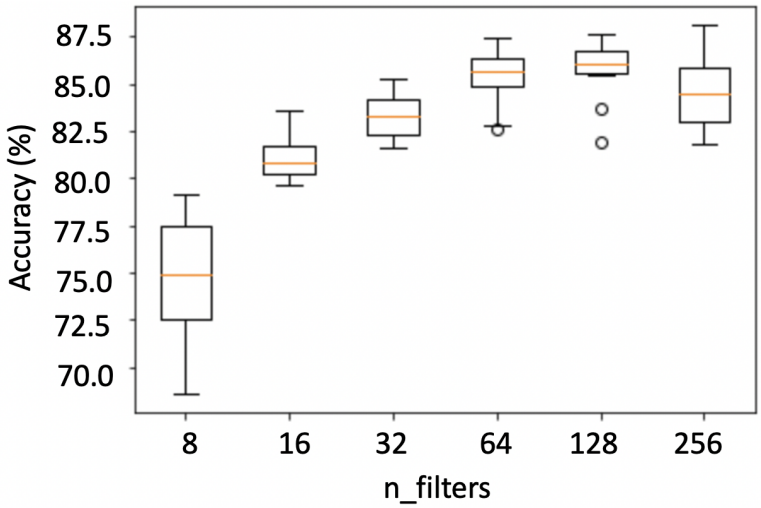

Figure 4.11: Accuracy for different number of filter maps, obtained for the second architecture in the UCI HAR database. The best configuration, in terms of stability and computational time is given by a number of filters equal to 64.

Setting the number of features maps to 64, the tuning of the kernel size obtained the following results, represented in the **Figure 4.12**.

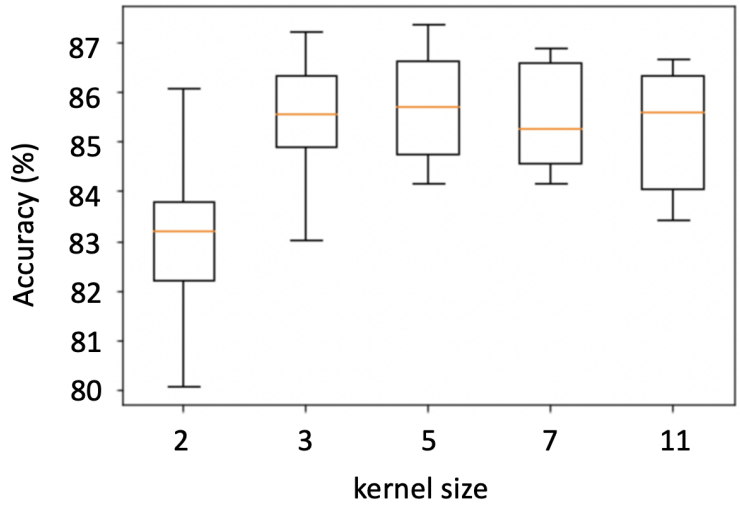

Figure 4.12: Accuracy for different values of the kernel size for the second architecture in UCI HAR database. Setting n = 64 feature maps, the best configuration is given by a value of 5.

In the **Figure 4.12** can be observed a general increase in model performance when increasing the kernel size. Results suggest that a kernel size of 5 represents the best configuration, providing accuracy of 85.7% with a standard deviation of ±1.0.

Setting the best configurations chosen previously, the CNN model extracted a number of parameters equal to 975620, as showed in the **Figure 4.13**.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 124, 64)           1984
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 62, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 58, 64)            20544
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 29, 64)            0
_________________________________________________________________
dropout (Dropout)            (None, 29, 64)            0
_________________________________________________________________
batch_normalization (BatchNo (None, 29, 64)            256
_________________________________________________________________
flatten (Flatten)            (None, 1856)              0
_________________________________________________________________
dense (Dense)                (None, 512)               950784
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 2052
=================================================================
Total params: 975,620
```

Figure 4.13: CNN model's second architecture in UCI HAR dataset. Using a number of filters equal to 64 and a kernel size of 5 led to a total a number of parameters equal to 975620.

The accuracy trend on the train and validation set are reported in the **Figure 4.14**, where it can be observed how the accuracy of the train set increases gradually while increasing the number of epochs, achieving the highest value in the last epoch, where the value is about 85%. The accuracy trend on the validation set is similar to that of the training set, despite a dip is evident in the 2_{nd} epoch.

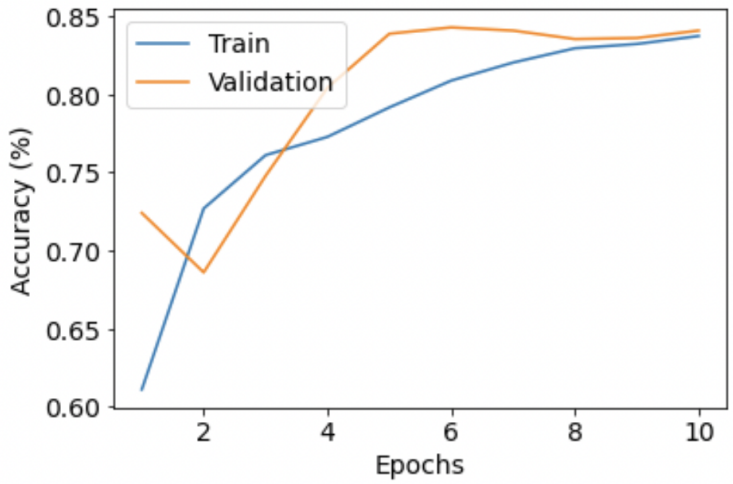

Figure 4.14: Accuracy trend on the train and validation set for the 2^{nd} architecture in the UCI HAR database.

Performances computed on the test set are summarized in the **Figure 4.15**.

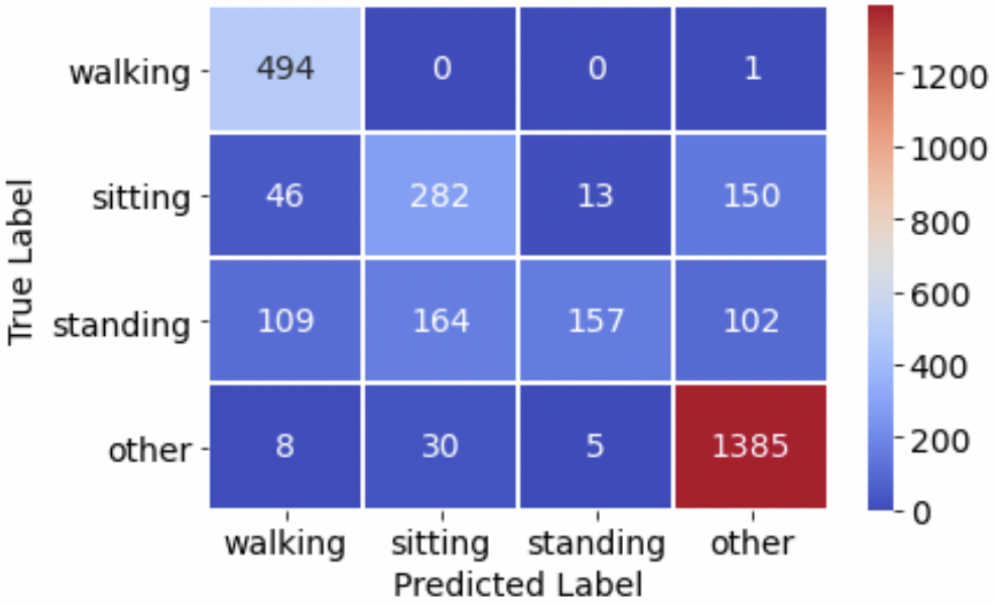

Figure 4.15: Confusion matrix computed on the test set of UCI HAR database in the second architecture.

Metrics computed on the confusion matrix of the **Figure 4.15** are listed in the **Table 4.6**

activity	accuracy(%)	specificity(%)	precision(%)	sensitivity(%)	f_1-score(%)
walking	94.4	93.3	75.1	99.7	85.7
sitting	86.3	92.0	59.2	57.4	58.3
standing	86.6	99.2	89.7	29.5	44.4
other	89.9	83.3	84.5	96.9	90.3

Table 4.6: Metrics calculated from the confusion matrix on the test set of the UCI HAR database in the second architecture.

From the results it is evident that performance are high, except for recall and f_1-score of 'sitting' and 'standing'.
From classes prediction on HAPT database the confusion matrix reported in the **Figure 4.16** was obtained.
Metrics computed on HAPT dataset are listed in the **Table 4.7**.
Figure 4.16 and **Table 4.7** show that better performances are obtained for 'walking' activity, where performances are about 80% on average. Moreover, the model allows to achieve the highest specificity and precision for 'sitting', because the number of false positive is equal to 0, despite sensitivity and f_1-score are very low for this class.
Performances obtained on 'other' class are very low, because activities belonging to this class are confused with 'walking'. This may be caused by the fact that 'other' include activities such as 'climbing down' and 'climbing up', that are very similar to 'walking'.

Figure 4.16: Confusion matrix from classes prediction on HAPT dataset.

activity	accuracy(%)	specificity(%)	precision(%)	sensitivity(%)	f_1-score(%)
walking	75.3	83.3	89.3	71.2	79.3
sitting	84.6	100.0	100.0	0.9	1.9
standing	87.7	89.7	60.8	77.7	68.3
other	73.5	74.4	0.1	3.2	0.3

Table 4.7: Metrics computed from the confusion matrix obtained from classes prediction on HAPT dataset.

Similarly, classes prediction on the RealWorld HAR dataset led to the confusion matrix reported in the **Figure 4.17**.

Figure 4.17: Confusion matrix from classes prediction on RealWorld HAR dataset.

All metrics computed from the **Figure 4.17** are listed in the **Table 4.8**.

activity	accuracy(%)	specificity(%)	precision(%)	recall(%)	f_1-score(%)
walking	64.6	59.2	32.0	90.1	47.2
sitting	83.6	99.0	70.8	10.5	18.4
standing	85.0	88.3	55.2	69.1	61.4
other	61.1	83.2	66.9	37.0	47.6

Table 4.8: Metrics calculated from the confusion matrix obtained from classes prediction on Real-World HAR dataset.

As can be observed from results, 'sitting' is wrongly predicted as 'standing' and 'other' is wrongly predicted as 'walking'. Results suggest the model isn't able to make correct classes prediction on the RealWorld HAR database.

3. *RealWorld HAR database's results*
   Mean accuracy on train, validation and test set, obtained in RealWorld HAR for each architecture are listed in the **Table 4.9**.

RealWorld HAR database			
	mean accuracy (%)		
architecture	**train**	**validation**	**test**
1C+ReLU-1P	94.8	95.5	79.3
1C+ReLU-1P-1C+RelU-1P	94.9	95.9	70.6
1C+ReLU-1P-1C+RelU-1P-1C+RelU-1P	94.6	95.7	72.8
2C+ReLU-1P	95.8	95.9	75.2
2C+ReLU-1P-1C	95.7	96.1	69.7

Table 4.9: Mean accuracy(%) on train,validation and test for each architecture of the CNN model in RealWorld HAR database.

Table 4.9 suggests the best architecture in terms of highest validation accuracy is the 5^{th} architecture.
Tuning the number of filters of the fifth architecture, the box-plot showed in the **Figure 4.18** was obtained. From the box-plot, it can be observed an increasing trend accuracy until to 64 feature maps with increasing the number of feature maps. The most stable configuration is given by a number of filter maps equal to 64, that led an accuracy of 97.03% with a standard deviation of ± 0.3.

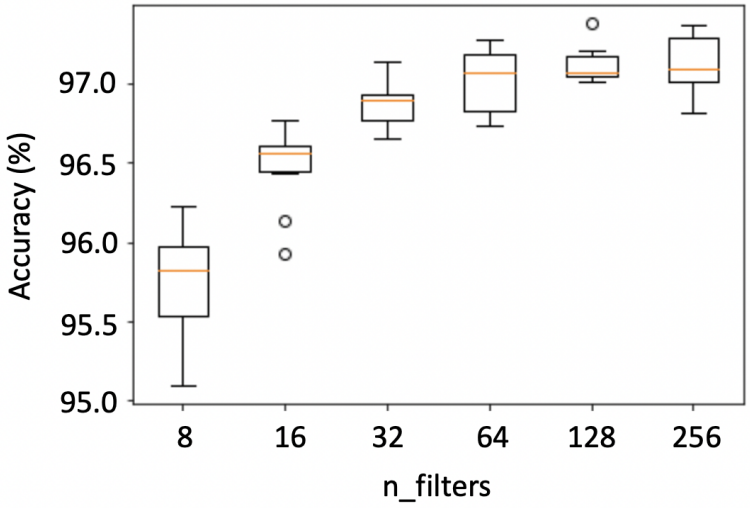

Figure 4.18: Accuracy for different values of the number filter maps, for the fifth architecture in RealWorld HAR database. The best configuration is 64.

Using n=64 feature maps, the tuning of the kernel size led the following results, as reported in the **Figure 4.19**.

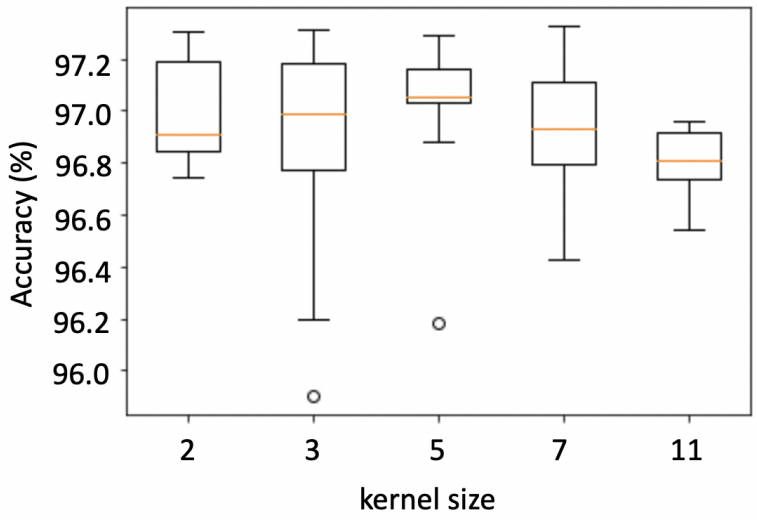

Figure 4.19: Accuracy for different values of the kernel size, for the fifth architecture in the RealWorld HAR database. Setting 64 feature maps, the best configuration is given by a value of 5.

Figure 4.19 shows the best configuration is a kernel size of 5, which provides an accuracy of 97.0% with a standard deviation of ±1.7.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 124, 64)           1984
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 120, 64)           20544
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 60, 64)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 56, 64)            20544
_________________________________________________________________
dropout (Dropout)            (None, 56, 64)            0
_________________________________________________________________
batch_normalization (BatchNo (None, 56, 64)            256
_________________________________________________________________
flatten (Flatten)            (None, 3584)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1835520
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 2052
=================================================================
Total params: 1,880,900
```

Figure 4.20: CNN model's fifth architecture in RealWorld HAR dataset: a number of filter set to 64 and a kernel size of 5 extract a number of parameters equal to 1880900.

Setting 64 feature maps and a kernel size of 5 the model extracts a total number of parameters equal to 1880900, as reported in the **Figure 4.20**.
Accuracy trends on train and validation set are depicted in the **Figure 4.21**, where it is evident the progressive increase of the accuracy on train set with increasing the number of epochs. The validation accuracy trend is similar to that of train set and assumes value between 94% and 96%.

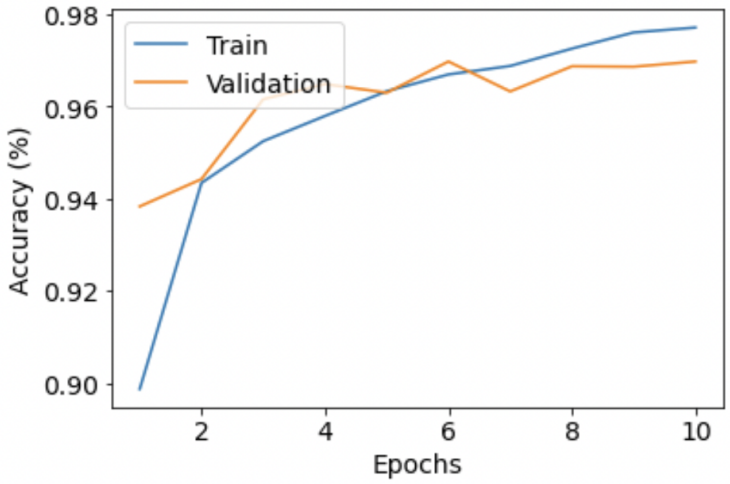

Figure 4.21: Accuracy trend on train and validation set for the 5^{th} architecture in the RealWorld HAR dataset.

Performances on the test set of the fifth architecture in the RealWorld HAR database are described in the **Figure 4.22**. Results of the **Figure 4.22** suggest performances on the

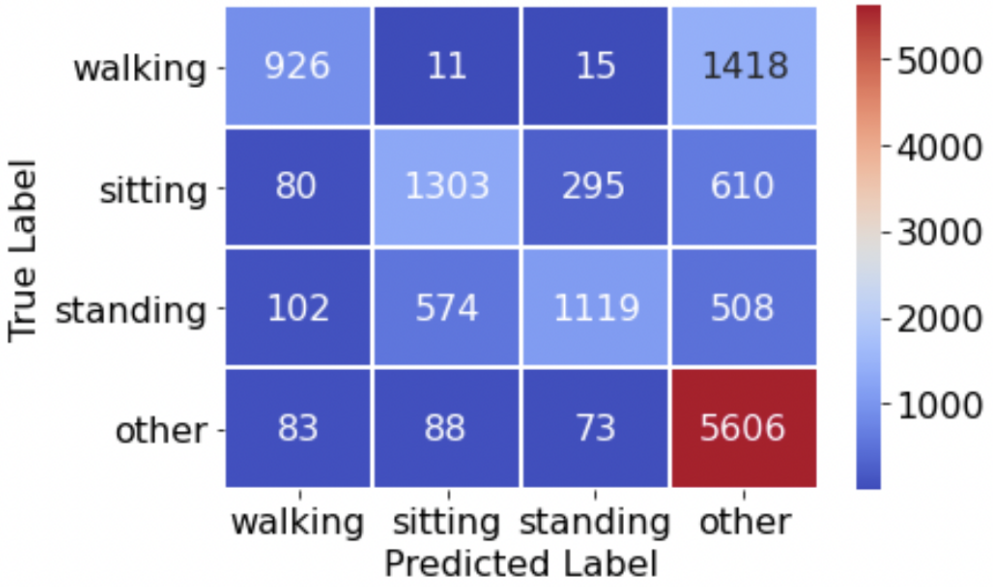

Figure 4.22: Confusion matrix calculated on the test set of RealWorld HAR database in the fifth architecture.

test are quite high, but are significantly smaller than that on train and validation set. For this reason, LOSO technique was applied and the results obtained from it are described in the **Figure 4.23**.

Figure 4.23: Confusion matrix calculated on the test set of RealWorld HAR database in the fifth architecture.

Metrics obtained from Hold-out and LOSO technique are compared in the **Table 4.10**. Comparing performances on test set from Hold-out technique and that from LOSO technique, reported in the **Table 4.10**, it is evident LOSO technique allows to achieve better performances of a value of 5%.

Activity	Validation	Performances on TestSet (%)				
		accuracy	specificity	precision	sensitivity	f_1-score
walking	HOLD OUT	86.6	97.4	77.7	39.0	52.0
	LOSO	**89.5**	95.2	73.8	**62.8**	**67.8**
sitting	HOLD OUT	87.0	93.6	65.9	56.9	61.1
	LOSO	**88.0**	**96.4**	**74.2**	48.3	58.5
standing	HOLD OUT	87.7	96.3	74.5	48.5	58.8
	LOSO	86.2	90.5	59.0	**65.6**	**62.2**
other	HOLD OUT	78.3	63.5	68.8	95.8	80.1
	LOSO	**86.5**	**80.5**	**81.5**	93.0	**86.9**

Table 4.10: Metrics obtained from Hold-out and LOSO technique for the fifth architecture in Real-World HAR database.

Similarly, performances obtained from classes prediction on HAPT database are reported in the **Figure 4.24**. All metrics computed from the confusion matrix of the **Figure 4.24** are listed in the **Table 4.11**.

Figure 4.24: Confusion matrix obtained from classes prediction on HAPT database.

As can be observed from the **Figure 4.24** and from the **Table 4.11** performances drop occurs in the prediction on HAPT database. The model isn't able to predict correctly 'walking' , as reported from performances very low, because it is wrongly predicted as 'other' classes. Summarizing, the model has mainly low performances.

activity	accuracy(%)	specificity(%)	precision(%)	recall(%)	f_1-score(%)
walking	53.0	90.9	15.7	7.9	10.5
sitting	88.4	94.3	50.7	43.8	47.0
standing	99.9	102.3	126.2	82.3	99.7
other	29.3	41.0	1.4	2.0	1.6

Table 4.11: Metrics calculated from the confusion matrix obtained from classes prediction on HAPT database.

Likewise, classes prediction on UCI HAR dataset led to the confusion matrix showed in the **Figure 4.25**.
Performances obtained testing the model on the UCI HAR dataset are low, especially precision,sensitivity and f_1 score. Accuracy and specificity are slightly higher, but don't exceed 72% on average. Totally, the performances are not high a lot.

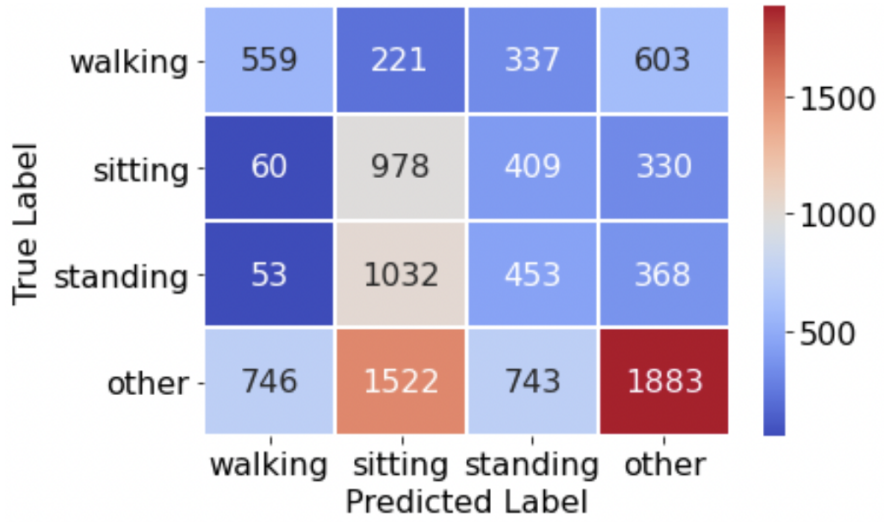

Figure 4.25: Confusion matrix obtained from classes prediction on UCI HAR dataset.

All metrics computed from the confusion matrix of the **Figure 4.25** are listed in the **Table 4.12**.

activity	accuracy(%)	specificity(%)	precision(%)	recall(%)	f_1-score(%)
walking	80.3	89.9	39.4	32.5	35.6
sitting	65.2	67.4	26.0	55.0	35.3
standing	71.4	82.2	23.3	23.7	23.5
other	58.1	75.9	59.1	38.4	46.6

Table 4.12: Metrics calculated from the confusion matrix obtained from classes prediction on UCI HAR dataset.

Table 4.13 summarizes the mean accuracy and f_1-score on the test set for each dataset.

TRAIN	Performances (%)	TEST		
		HAPT	UCI HAR	RealWorld HAR
HAPT	accuracy	**98.5**	66.5	69.0
	f_1-score	**91.9**	36.5	50.1
UCI HAR	accuracy	80.2	**89.3**	73.5
	f_1-score	37.4	**69.6**	43.6
RealWorld HAR	accuracy	67.6	68.7	**84.9**
	f_1-score	39.7	35.2	**63.9**

Table 4.13: Summary of mean accuracy and f_1-score on the test set for each dataset.

As can be observed from the **Table 4.13**, when the CNN model is trained on the train set of a database and tested on the own test set, performances obtained are high. On the other hand, when the model is used to make prediction on the other two databases, performances decline significantly.

Finally, the computational time (s) was evaluated both on the train and test set, including standardization, segmentation, training/validation and testing process. The computation time was computed for a duration of signal ranging from 10 to 100000 s. Results are reported in the following figures.

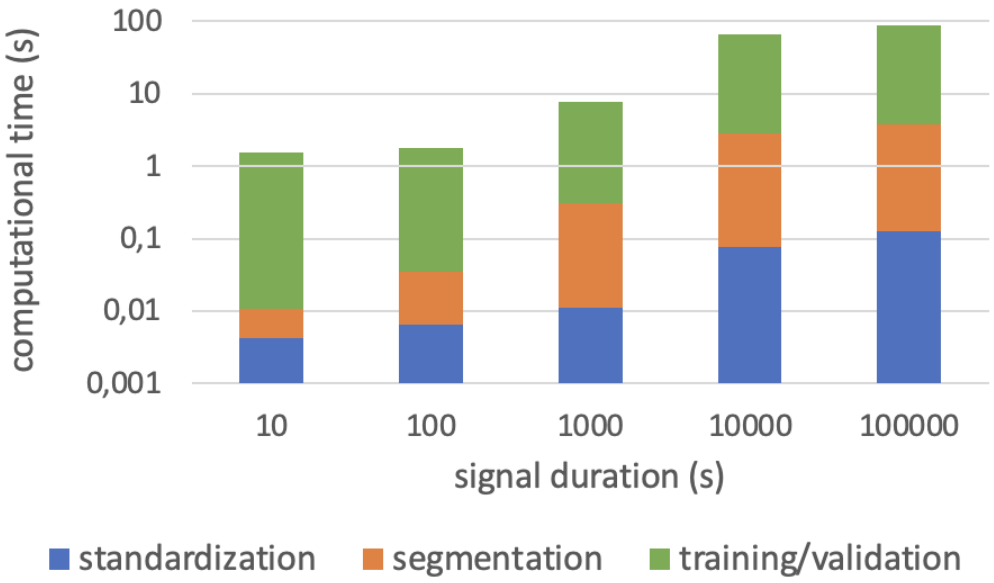

Figure 4.26: Computational time computed on standardization, segmentation and training/validation on the train set.

From the **Figure 4.26**, it is evident the time to compute training/validation process is more longer than that on standardization and segmentation. Standardization process requires few tenths of seconds, segmentation process needs of few seconds and training/validation requires some hundreds of seconds, corresponding around to one minute.

Figure 4.27 reports the total duration including standardization, segmentation and training/validation and suggests the trend of training/validation is very similar to that of total duration. This means that training/validation is the process with more computational time.

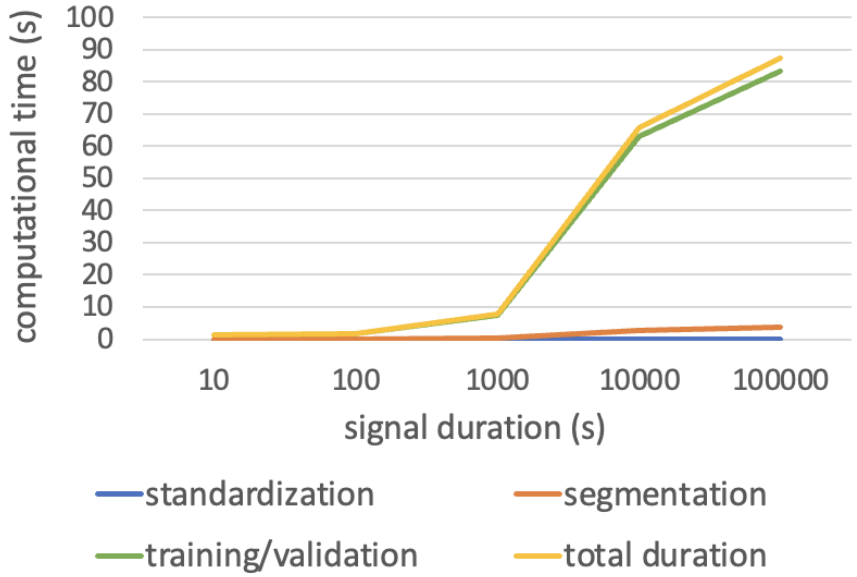

Figure 4.27: Computational time computed on standardization, segmentation and training/validation on the train set.

Likewise, computational time was computed on the test set.
Figure 4.28 reports the time(s) of standardization, segmentation and testing process on the test set. **Figure 4.28** suggests the time to compute standardization is few tenths of seconds,

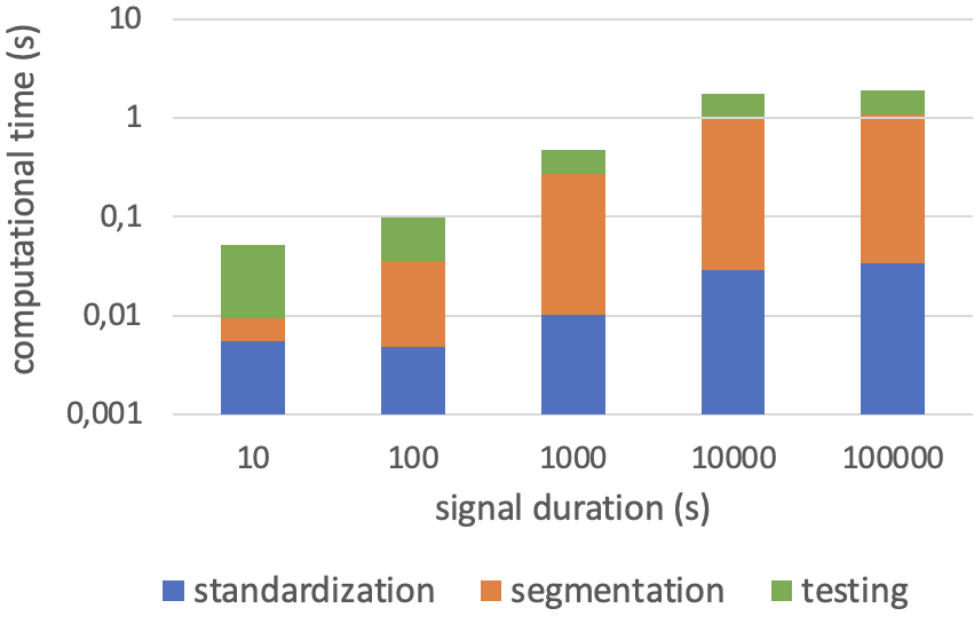

Figure 4.28: Computational time computed on standardization, segmentation and testing on the test set.

the time to compute segmentation is less of one second and that to compute testing is of the same order of segmentation process.
Figure 4.29 reports the trend of the processes and includes the total duration trend.

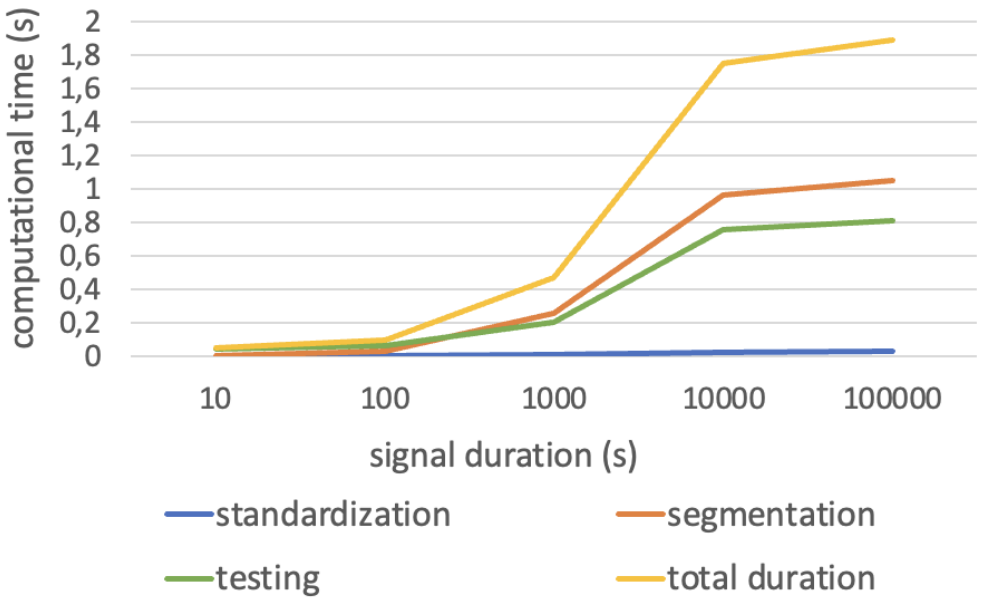

Figure 4.29: Computational time computed on standardization, segmentation and testing on the test set.

Figure 4.29 suggests the time to compute testing is very small, unlike the training/validation process.

Chapter 5

Discussion

This section presents and discusses the results of the current work with the aim of selecting the most effective points. To provide an exhaustive performance evaluation of the CNN model, five architectures were performed on the three selected databases; then the best architecture, in terms of performances, was chosen for each database. However, results obtained aren't immediately comparable with the results found in literature. In this work, in fact it was decided to select a smaller number of activities than literature and complex activities, such as jumping, climbing up, climbing down were rejected. This work selected only simple activities, which generally are activities performed by all subjects, from older to younger. This issue may impact on performances differences obtained from the proposed CNN model and on the performances obtained in the literature.
In addition, the duration of performed activities can adversely affect the performance of the model, as some activities rare performed for longer than others and this can affect the model's ability to discriminate between activities.
Concerning the performances on each database, it can be observed that the optimized model obtained an high classification rate in the recognition of ADLs, both on validation and on test set, despite the differences in performance being significantly evident between the datasets. The performances computed on the test set of the HAPT data have values similar to those of train and validation and are around 95%. The performances on the test set of the UCI HAR database are similar to those on train and validation and are around 76 %. While, on the test set of the RealWorldHAR, a drop in performances is significant. If the performances on train and validation are around 95%, those on test set are around 75%. This decline could be caused by the presence of over-fitting.
The LOSO technique allowed to improve performances on the test set, achieving an accuracy of 87.5% on average and a specificity of 90.6%. Better performances were obtained in particular on sensitivity and f_1 -score. The sensitivity went from an average value of 60.0% to 67.4%, while the score of f_1 increased from 62.9% to 68.8%.
On the other hand, the performance obtained by the model trained on its own train set and used to make predictions on other datasets led to worse performance. More specifically, the model trained on the train set of the HAPT dataset achieved performance of about 52% on average on the UCI HAR dataset and 60% on RealWorld HAR. The model trained on the train set of UCI HAR dataset resulted in approximately 58% performance on average on both the HAPT dataset and the RealWorld HAR dataset. While, the model trained on RealWorldHAR's train set and tested on two other datasets resulted performances of approximately 54% and 52% on average on the HAPT and the UCI HAR dataset respectively.
A possible performance degradation occurred when the model was trained on the train set of a dataset consisting of only 4 activities and used to predict classes on a dataset consisting of 6 activities.

In addition to the classification performance, the computational complexity was calculated, reporting the total number of extracted parameters. More complex the architecture and more parameters are extracted. As a result, the computation time to run the algorithm increases. For this reason the choice of the best architecture of each architecture was the compromise between performance and computational complexity.
However, a possible question arises from this work: *"Is it possible to select a common architecture for the three databases, considering that the performances are more or less similar to each other?"*
Perhaps the 2^{nd} architecture can be chosen as the common architecture, because it is the fastest architecture and allows for great performance. If the 2^{nd} architecture is selected in the HAPT database, the performance drops by only 0.4%. While, if you select the 2^{nd} architecture in the RealWorld HAR database, the performance decreases by only 0.2%.
Another issue to point out is the difference in computation time required to perform standardization, segmentation, training/validation and testing processes. Standardization and segmentation are two processes that take little time to execute, a few tenths of a second for standardization and a few seconds for segmentation. The training / validation process, on the other hand, takes about one minute, for about 28 hours of activity. However, once the training process is finished, the testing process is very fast and takes only tenths of a second.

Chapter 6

Conclusions and Future Works

In this work, sensor-based HAR was used to predict and identify classes of a variety of ADIs on subjects wearing a waist-mounted smartphone integrated with two wearable sensors: accelerometer and gyroscope. Sensor data was acquired from the three most common databases used in human activity research fields: HAPT, UCI HAR, and RealWorld HAR database, from which only simple activities were selected.
The proposed model was the CNN model, built layer by layer, in which the tuning of the number of layers, feature maps and kernel size were performed.
Hyper-parameters tuning led to choosing the best configuration model for each database, in terms of performance and computational complexity. The optimized model trained on the train set, evaluated on the validation set and tested on the test set brought great performance in each database: about 95% on the HAPT dataset and 80% on the UCI HAR dataset. On the RealWorld HAR dataset, performance reached high values on the train and validation set, but declined on the test set. The LOSO technique allowed to improve the performance on the test set, in particular the sensitivity and the f_1-score, which increased by about 10%. However, the performance obtained from the model trained on its own train set and used to predict classes on the other two entire datasets had a significant drop. Accuracy decreased by approximately 30% when the model was trained on the train set of the HAPT database, while the f_1 score decreased by approximately 53%. The same problem occurs in the UCI HAR dataset, where the precision decreased by approximately 14 % and the f_1 score decreased by approximately 42%. In the RealWorld HAR dataset, the accuracy decreased by approximately 20%, while the f_1 score decreased by approximately 41%.
The performance demonstrated the CNN model's ability to make correct predictions of classes on each database's test set and suggested the model's inability to make predictions on other databases.
As future works, in order to get better performance on the test set and try to achieve similar performance over activities, that performing for a longer time may be downsampled, while less performed tasks may be oversampled.
Additionally, the CNN model could be used to support early diagnosis and treatment of PD: the CNN model could be trained on an online dataset and tested on a dataset, containing data from older subjects or PD.
Finally, Transfer Learning could be another improvement to apply. The pre-trained model could be used as a starting point for training the model on new data.

Bibliography

[1] L. Minh Danga, Kyungbok Mina, Hanxiang Wanga, Md. Jalil Pirana, Cheol Hee Leeb, Hyeonjoon Moona, *Sensor-based and vision-based human activity recognition: A comprehensive survey* , Department of Computer Science and Engineering, Sejong University, Seoul 143-747(05006), South Korea, Deep inspection, A-1314, Beopwon-r o 11 gil 25, Songpa-gu, Seoul 143-747(05006), South Korea

[2] Khan, W.Z.; Xiang, Y.; Aalsalem, M.Y.; Arshad, Q. *Mobile phone sensing systems: A survey.* in IEEE Commun. Surv. Tutor. 2013, 15, 402–427.

[3] Florenc Demrozi, Cristian Turetta, Graziano Pravadelli, *B-HAR: an open-source baseline framework for in depth study of human activity recognition datasets and workflows*,IEEE Access

[4] Daniela Micucci ,Marco Mobilio andPaolo Napoletano, *UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones*, Department of Informatics, System and Communication, University of Milano

[5] Shoaib, M.; Bosch, S.; Incel, O.D.; Scholten, H.; Havinga, P.J, *Complex human activity recognition using smartphone and wrist-worn motion sensors*, Sensors 2016, 16, 426.

[6] Figo, D.; Diniz, P.C.; Ferreira, D.R.; Cardoso, J.M.P. *Preprocessing techniques for context recognition from accelerometer data.*, Pers. Ubiquitous Comput. 2010, 14, 645–662.

[7] Diederik P. Kingma, Jimmy Ba, *Adam: A Method for Stochastic Optimization*, Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015

[8] Jason Brownlee, *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*, December 3, 2018 in Deep Learning Performance

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, 2014.

[10] Sergey Ioffe, Christian Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*

[11] Andrey Ignatov, *Real-time human activity recognition from accelerometer data using Convolutional Neural Networks*,Swiss Federal Institute of Technology in Zurich (ETHZ), Switzerland

[12] Fontecha J, Navarro F.J, Hervàs R,Bravo J, *Elderly frailty detection by using accelerometer-enabled smartphones and clinical information records.*, Pers. Ubiquitous Comput. 2013, 7, 1073–1083.

[13] Sousa, W.; Souto, E.; Rodrigres, J.; Sadarc, P.; Jalali, R.; El-khatib, K. *A Comparative Analysis of the Impact of Features on Human Activity Recognition with Smartphone Sensors.*, In Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web, Gramado, RS, Brazil, 17–20 October 2017; pp. 397–404

[14] Miluzzo, E.; Lane, N.D.; Fodor, K.; Peterson, R.; Lu, H.; Musolesi, M.; Eisenman, S.B.; Zheng, X.; Campbell, A.T. *Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application.*, In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, Raleigh, NC, USA, 5–7 November 2008; pp. 337–350.

[15] Lane, N.; Mohammod, M.; Lin, M.; Yang, X.; Lu, H.; Ali, S.; Doryab, A.; Berke, E.; Choudhury, T.; Campbell, A. BeWell, *A Smartphone Application to Monitor, Model and Promote Wellbeing*, In Proceedings of the 5th International ICST Conference on Pervasive Computing Technologies for Healthcare, Dublin, Ireland; 2011; p. 8.

[16] B. Alsinglawi, Q.V. Nguyen, U. Gunawardana, A. Maeder, S.J. Simoff, *Rfid systems in healthcare settings and activity of daily living in smart homes: a review.*, E-Health Telecommun. Syst. Netw. 6 (2017) 1–17.

[17] Thiemjarus, S.; Henpraserttae, A.; Marukatat, S. *A study on instance-based learning with reduced training prototypes for device-context-independent activity recognition on a mobile phone*, In Proceedings of the 2013 IEEE International Conference on Body Sensor Networks, BSN, Cambridge, MA, USA, 6–9 May 2013

[18] Hynes, M.; Wang, H.; McCarrick, E.; Kilmartin, L. *Accurate monitoring of human physical activity levels for medical diagnosis and monitoring using off-the-shelf cellular handsets.*, Pers. Ubiquitous Comput. 2011, 15, 667–678.

[19] Mladenov, M.; Mock, M., *A step counter service for Java-enabled devices using a built-in accelerometer.*, in Proceedings of the 1st International Workshop on Context-Aware Middleware and Services affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009), Dublin, Ireland, 16 June 2009.

[20] Zhang, S.; McCullagh, P.; Nugent, C.; Zheng, H., *Activity Monitoring Using a Smart Phone's Accelerometer with Hierarchical Classification.*, in Proceedings of the 2010 Sixth International Conference on Intelligent Environments, Kuala Lumpur, Malaysia, 19–21 July 2010; pp. 158–163.

[21] Yang, J. *Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones.*, in Proceedings of the 1st International Workshop on Interactive Multimedia for Consumer Electronics, Beijing, China, 23 October 2009; pp. 1–9.

[22] Schindhelm, C.K. *Activity recognition and step detection with smartphones: Towards terminal based indoor positioning system.*, In Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, Sydney, NSW, Australia, 9–12 September 2012; pp. 2454–2459.

[23] Konstantinos Peppas,Apostolos C. Tsolakis, Stelios Krinidis and Dimitrios Tzovaras, *Real-Time Physical Activity Recognition on Smart Mobile Devices Using Convolutional Neural Networks*, Centre for Research and Technology, Hellas, Information Technologies Institute

[24] Wesllen Sousa Lima,Eduardo Souto, Khalil El-Khatib, Roozbeh Jalali and Joao Gama, *Human Activity Recognition Using Inertial Sensors in a Smartphone: An Overview*, Universidade Federal do Amazonas, Manaus 69080-900, Brazil, University of Ontario Institute of Technology, Oshawa ON L1H 7K4, Canada, Institute for Systems and Computer Engineering, Technology and Science—INESCTEC, Portugal

[25] Henry Friday Nweke, Ying Wah Teh , Mohammed Ali Al-garadi, Uzoma Rita Alo, *Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges*, Department of Information Systems, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia b Computer Science Department, Ebonyi State University, Abakaliki, Ebonyi State P.M.B 053, Nigeria

[26] Kun Wang; Jun He; Lei Zhang, *Attention-Based Convolutional Neural Network for Weakly Labeled Human Activities' Recognition With Wearable Sensors*,School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, China

[27] Cook, D.J.; Krishnan, C.N. *Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data*, Hoboken, NJ, USA, 2015.

[28] S Tsokov, M Lazarova and A Aleksieva-Petrova, *Accelerometer-based human activity recognition using 1D convolutional neural network*, Technical University of Sofia, Faculty of Computer Systems and Technologies, Department Computer Systems, 8 Kliment Ochridsky blvd., Sofia 1000, Bulgaria

[29] Mekruksavanich, S.; Hnoohom, N.; Jitpattanakul, *A. Smartwatch-based sitting detection with human activity recognition for office workers syndrome.*, In Proceedings of the 2018 International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON), Muang Chiang Rai, Thailand, 25–28 February 2018; pp. 160–164.

[30] Y. Tang, Q. Teng, L. Zhang, F. Min and J. He, *"Layer-Wise Training Convolutional Neural Networks With Smaller Filters for Human Activity Recognition Using Wearable Sensors"*, in IEEE Sensors Journal, vol. 21, no. 1, pp. 581-592, 1 Jan.1, 2021, doi: 10.1109/JSEN.2020.3015521.

[31] I. Portugal, P. Alencar, D. Cowan, *The use of machine learning algorithms in recommender systems: a systematic review.*, Expert Syst. Appl. 97 (2018) 205–227.

[32] Jindong Wanga,b, Yiqiang Chena,b, Shuji Haoc, Xiaohui Penga,b, Lisha Hua, *Deep learning for sensor-based activity recognition: A survey*, Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, University of Chinese Academy of Sciences, Beijing, China, Institute of High Performance Computing, Singapore

[33] Choujaa, D.; Dulay, N. *Activity Recognition from Mobile Phone Data: State of the Art.*, Prospects and OpenProblems. Imp. Coll. Lond. 2009, 5, 32.

[34] Chen, L.; Hoey, J.; Nugent, C.D.; Cook, D.J.; Yu, Z.; Member, S. *Sensor-Based Activity Recognition.*, Syst. Man Cybern. Part C Appl. Rev. 2012, 42, 790–808.

[35] Incel, O.D.; Kose, M.; Ersoy, C. *A Review and Taxonomy of Activity Recognition on Mobile Phones.*, BioNanoScience 2013, 3, 145–171.

[36] Khusainov, R.; Azzi, D.; Achumba, I.E.; Bersch, S.D. *Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations.*, Sensors 2013, 13, 12852–12902.

[37] Avci, A.; Bosch, S. *Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey.*, In Proceedings of the 23th International conference on architecture of computing systems (ARCS), Hannover, Germany, 22–23 February 2010; pp. 1–10.

[38] Chen, Y.; Shen, C. *Performance Analysis of Smartphone-Sensor Behavior for Human Activity Recognition.*, IEEE Access 2017, 5, 3095–3110.

[39] Wang, J.; Chen, Y.; Hao, S.; Peng, X.; Hu, L. *Deep Learning for Sensor-based Activity Recognition: A Survey.*,Comput. Vis. Pattern Recognit. 2017, 119, 3–11.

[40] Li, F.; Shirahama, K.; Nisar, M.; Köping, L.; Grzegorzek, M, *Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors.*, Sensors 2018, 18, 679.

[41] Zahraa S. Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. *Activity Recognition with Evolving Data Streams: A Review.*, ACM Comput. Surv. 51, 4, Article 71 (September 2018), 36 pages. DOI:https://doi.org/10.1145/3158645

[42] Pichao Wang, Wanqing Li, Philip Ogunbona, Jun Wan, Sergio Escalera, *RGB-D-based human motion recognition with deep learning: A survey*,Computer Vision and Image Understanding, Volume 171, 2018, Pages 118-139,ISSN 1077-3142,

[43] O.D. Lara, M.A. Labrador, *A survey on human activity recognition using wear- able sensors.*, IEEE Commun. Surv. Tutor. 15 (3) (2012) 1192–1209.

[44] Florenc Demrozi, Graziano Pravadelli,Azra Bihorac and Parisa Rashidi. *Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey*, Department of Computer Science, University of Verona, 37134 Verona, Italy, Division of Nephrology, Hypertension, and Renal Transplantation, College of Medicine, University of Florida, Gainesville, FL 32610, USA, Department of Biomedical Engineering, University of Florida, Gainesville, FL 32610, USA

[45] A. Krizhevsky, I. Sutskever, G.E. Hinton, *ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems.*, 2012, pp. 1097–1105.

[46] Fernando Moya Rueda; ID , René Grzeszick ; Gernot A. Fink ; Sascha Feldhorst and Michael ten Hompel, *Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors*,Department of Computer Science, TU Dortmund University, 44227 Dortmund, Germany; Department of Mechanical Engineering, TU Dortmund University, 44227 Dortmund, Germany

[47] T. Zebin, P. J. Scully, N. Peek, A. J. Casson and K. B. Ozanyan *Design and Implementation of a Convolutional Neural Network on an Edge Computing Smartphone for Human Activity Recognition* in IEEE Access, vol. 7, pp. 133509-133520, 2019, doi: 10.1109/ACCESS.2019.2941836.

[48] D. Gholamiangonabadi, N. Kiselov and K. Grolinger, *Deep Neural Networks for Human Activity Recognition With Wearable Sensors: Leave-One-Subject-Out Cross-Validation for Model Selection*, in IEEE Access, vol. 8, pp. 133982-133994, 2020, doi: 10.1109/ACCESS.2020.3010715.

[49] R. Xi, M. Li, M. Hou, M. Fu, H. Qu, D. Liu, and C. R. Haruna, emphDeep dilation on multimodality time series for human activity recognition,,IEEE Access, vol. 6, pp. 53381–53396, 2018.

[50] Oresti Banos, Juan-Manuel Galvez, Miguel Damas, Hector Pomares and Ignacio Rojas *Window Size Impact in Human Activity Recognition*, Department of Computer Architecture and Computer Technology, Research Center for Information and Communications Technologies—University of Granada (CITIC-UGR), C/Calle Periodista Rafael Gomez Montero 2, Granada E18071, Spain

[51] J.Liono,A.K.Qin,andF.D.Salim, *Optimal time window for temporal segmentation of sensor streams in multi-activity recognition*, in Proc. 13th Int. Conf. Mobile Ubiquitous Syst., Comput., Netw. Services, 2016, pp. 10–19.

[52] Banos,O.;Garcia,R.;Holgado,J.A.;Damas,M.;Pomares,H.;Rojas,I.;Saez,A.;Villalonga, C.mHealthDroid, *A novel framework for agile development of mobile health applications.*, in Proceedings of the 6th International Work-Conference on Ambient Assisted Living an Active Ageing (IWAAL 2014), Belfast, UK, 2–5 December 2014.

[53] Anguita,D.;Ghio,A.;Oneto,L.;Parra,X.;Reyes-Ortiz,J.L, *A Public Domain Dataset for Human Activity Recognition Using Smartphones*, In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 24–26 April 2013; pp. 24–26.

[54] Sikder, N.; Chowdhury, M.; Arif, A.; Nahid, *A. Human Activity Recognition Using Multichannel Convolutional Neural Network.*, in Proceedings of the 2019 5th International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, Bangladesh, 26–28 September 2019.

[55] Ronao, C.A.; Cho, S.B., *Human activity recognition with smartphone sensors using deep learning neural networks.* ,Expert Syst. Appl. 2016, 59, 235–244.

[56] Ferrari, D. Micucci, M. Mobilio, and P. Napoletano, *On the personalization of classification models for human activity recognition.*, IEEE Access, vol. 8, pp. 32066–32079, 2020.

[57] , M.-O.Mario, *Human activity recognition based on single sensor square HV acceleration images and convolutional neural networks.*, IEEE Sen- sors J., vol. 19, no. 4, pp. 1487–1498, Feb. 2019.

[58] *WISDM's activity prediction dataset* http://www.cis.fordham.edu/wisdm/ dataset.php.

[59] Lockhart, J.W.; Weiss, G.M.; Xue, J.C.; Gallagher, S.T.; Grosner, A.B.; Pulickal, T.T., *Design Considerations for the WISDM Smart Phone-based Sensor Mining Architecture.*, in Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data, San Diego, CA, USA, 21–24 August 2011; pp. 25–33.

Ringraziamenti

"Lo sport insegna che per la vittoria non basta il talento, ci vuole il lavoro e il sacrificio quotidiano. Nello sport come nella vita." (Pietro Mennea)

Questa frase riassume in pieno il mio percorso universitario, fatto di tanto impegno e tanti sacrifici, che mi hanno portato a questo fatidico giorno.

Desidero ringraziare la relatrice di questo lavoro, la Prof. Gabriella Olmo, per avermi dato l'opportunità di realizzare la mia Tesi di Laurea, nonostante il periodo storico complesso e instabile e per avermi indirizzato su una tematica molto affascinante, che spero di approfondire negli anni.

Un ringraziamento sincero all'Ing.Luigi Borzì, una guida indispensabile per la stesura di questa Tesi. Con la sua massima disponibilità, non scontata direi, mi ha fornito un supporto costante e ottimi suggerimenti.

Grazie Claudio per aver sempre appoggiato le mie scelte, per avermi supportato, sopportato, confortato e per aver gioito insieme a me. Grazie per essere il mio compagno di vita. La mia stima per te cresce ogni giorno sempre di più.

Infine, il ringraziamento più grande va alla mia famiglia per avermi sostenuta pazientemente, per aver sempre creduto in me e avermi dato la possibilità di costruire un futuro. GRAZIE. Senza di voi non sarei quella che sono. Siete voi la mia forza e il mio spirito guida.