Automatic Activity Recognition by means of Wearable Sensors and Supervised Learning Techniques

Fabián Andrés Gómez Meneses

Universidad Nacional de Colombia
Faculty of Engineering, Department of Systems and Computer Engineering
Bogotá, Colombia
2018

Automatic Activity Recognition by means of Wearable Sensors and Supervised Learning Techniques

Fabián Andrés Gómez Meneses

Final work presented for the degree of:
Master of Systems Engineering

Director:
Fabio A. González O., Ph.D. & M.Sc. in Computer Science, M.Sc. in Mathematics

Research line:
Deep learning
Research group:
MindLab

Universidad Nacional de Colombia
Faculty of Engineering, Department of Systems and Computer Engineering
Bogotá, Colombia
2018

Abstract

Human activity recognition (HAR) is at the forefront of *Pervasive Computing* efforts, and deep learning techniques currently empower the most successful endeavors within the field. By using a publicly available dataset an exploratory analysis of feature learning is put forward in this work. The convolutional neural network deployed here highlights both the advantages and limitations of this class of models, while offering an overview of machine learning-aided human behavior analysis. Furthermore, the exploration includes an experimental comparison with a more traditional SVM model with feature engineering, over the same data.

Keywords: machine learning, deep learning, human behavior, neural networks, pervasive computing)

Contents

Figure list

Table list

1 Introduction

Out of the many views meeting in human behavior analysis two rather different approaches retain special interest. First, from the computer science realm, the *interactional* sense of the human-computer relationship central to Ubiquitous Computing. And then, the powerful ethnographic description of human customs which anthropologists are known for (part of their encompassing account of human condition).

In 1991 Mark Weisser, a former researcher of the computer science laboratory at Xerox PARC, wrote "The Computer of the 21st Century"[3]. There Weisser introduced what soon became the navigation chart of *Pervasive* or Ubiquitous Computing: the now commonplace idea of computing interleaved with daily life. Processing, storage and portability restrictions back then have been replaced with ubiquitous technologies realizing that vision. However, transparency in the interaction between machine and human being remains a prevalent topic, with context as its core concept.

Context is, to this day, a highly complex and theoretical matter with ample review. In the center of the accounts of context is the *subject*, both as the source of interactions and as the main recipient of system adaptability. This subject from Ubiquitous Computing, taken as a concrete agent of actions, effectively connects the points of view mentioned above by means of its *observable* and *measurable* behavior.

This dimension of behavior becomes specially suited to systematic capture and analysis when a hierarchical perspective is introduced: simple gestures and motions by one person compose basic activities, and lead in turn to more complex activities, and to additional subjects getting involved. Empowered by motion sensing technologies, the field of Human Activity Recognition (HAR) embodies this view in its commitment to identify activities performed by subjects in their everyday interactions with the physical world.

With this background in place the question setting the scope for this work can be stated as *how to process sensor data in order to extract meaningful human activity information out from it?*. An attempt is made here to answer this question, by deploying a supervised and deep feature learning strategy. In particular, Convolutional Neural Networks (*CNNs* or *ConvNets*) are used to this end.

ConvNets do make sense for HAR and its raw signals because of their capability to process inputs with high dimensionality (i.e. the three axes of movement per sensor resemble RGB channels of an image). Moreover, CNNs are able to capture local dependencies (nearby inertial readings are likely to be correlated) and scale invariant features (different subjects are expected to perform basic motions at different paces). Also, these networks are computationally efficient, and do not require DSP expertise to start performing well over raw signals.

By building upon properties specific to CNNs, this work sets as its main goal to demonstrate the feasibility of a feature learning approach to HAR. In this sense, it seeks to extract meaningful representations of human activities from raw motion signals. In addition, the advantages of a feature learning approach will be highlighted, by comparing the proposed model to a more traditional SVM approach with feature engineering.

The Smartlab's HAR dataset, a publicly available dataset, will provide the experimental raw signals required to explore the proposed feature learning classifier for HAR. This specific dataset was released alongside experimental results of a more traditional SVM model, which therefore will serve as benchmark for the proposed model.

The main contributions of this work are:

- A feature learning model for HAR is implemented and evaluated against the raw signals in a public dataset, reaching performance scores on par with more traditional approaches based on expert feature design.
- An effort is documented to reproduce the process and results of an SVM classifier with handcrafted features input, showing the limitations of such methodology.
- The process of hyperparameter exploration carried out reveals the difficulties found in tuning a CNN model, especially when the initial performance is satisfactory.

The remaining of this document is organized as follows. Chapter 2 discusses past and current research within HAR field, encompassing more traditional studies and recent deep learning works. As part of this chapter a classification of activity recognition research is outlined. Chapter 3 presents an exploratory analysis of Smartlab's HAR dataset, with a detailed review of its contents. This section also registers an attempt to replicate the process and results of the SVM-based approach adopted by the dataset's authors. Chapter 4 introduces the proposed model for HAR, describing its building blocks and related architectural decisions. As for chapter 5 it includes the results from the experimental evaluation of two alternative implementations of the new HAR model. Here, the HAR dataset serves as input data and the performance of the CNN model is compared with the scores from the original SVM model. Finally, chapter 6 discusses findings and suggests a path to extend the scope of this work.

2 Related work

This chapter is the result of a systematic review of works related to Human Activity Recognition (HAR). The content is structured to start from the basic concept of activity with related sensing and recognition tasks, and then explores useful classifications of models and techniques. Afterwards, trends applying deep learning to HAR are presented. The final section lists public datasets available for HAR research, completing the landscape next chapters build upon.

2.1. Activity and related concepts

HAR's central concept, *activity*, differs from other ideas in Pervasive Computing in its down-to-earth character. When compared against an abstract and broad concept such as *context*, activities -taken as sequences of actions- are more related to observable human behavior. Also, from a low-level perspective activities show a closer resemblance to the data captured by sensors. For those reasons activity recognition is an approachable entry point to the basic understanding and modeling of human context.

The core idea behind an approach to context originating from activity is that a person's behavior is highly dependent on perception, environment, prior knowledge, and interaction with others. In this sense *activity* as a retrospective context-representation enabler requires first and foremost an operational concept.

2.1.1. Activity

Activity is a concept not frequently made explicit as part of studies taking behavioral patterns as research matter. Its meaning seems to be so closely related to daily experience that a proper conceptualization is often omitted. Nonetheless, in the articles reviewed a powerful idea links activity to context: Dourish describes context as a relational property in the interaction between objects or activities, transforming it into an emergent feature of activity [4]. This departs from the usual trend of placing context before activity, and further disables the centrality of context by requiring a previous static modeling of activities.

A more operational concept of activity can be found in [5] where a distinction between activity and action is proposed. Both terms may refer to human behavior but while action is

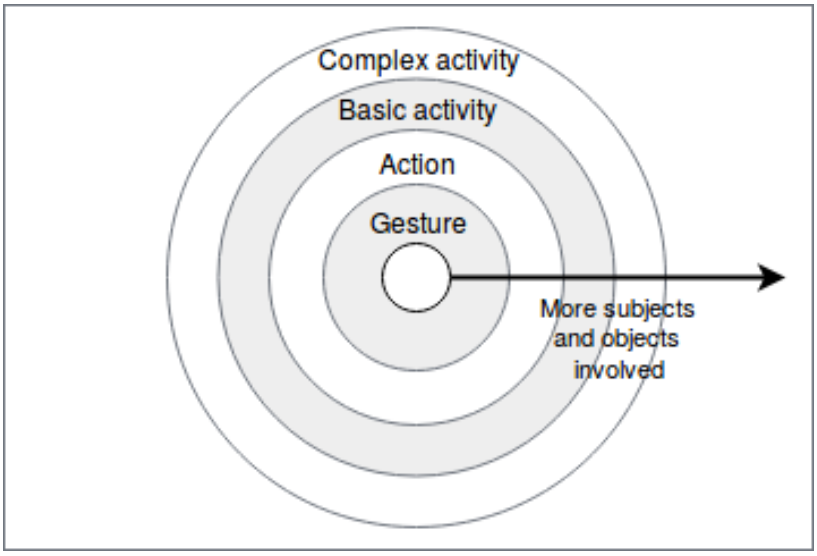

Figure 2-1: General hierarchy shared by concepts of activity

short-lived and executed by a single person, activity designates sequences of one or more actions performed by one or more subjects (with some degree of juxtaposition). A similar view is held in [6], where activities are either elementary (*motor words*) or composite (*motor sentences*), the later being sequences of elementary activities.

The computer vision and video surveillance fields also support a hierarchical view of activity (see figure **2-1**), with gradually increasing complexity and time lengths: *gestures* match elementary movements of body parts; *actions* are gestures orchestrated by a single person; *interactions* encompass human-human and human-objects activities; and *group activities* are executed by a set of persons and objects [7].

The health care community has long provided one of the most practical meanings of activity, as it differentiates activities of daily living (*ADLs*) from instrumental activities of daily living (*IADLs*), which involve interactions with instruments or appliances [8]. The current practice however is to use ADLs in a general manner, including IADLs. The ongoing centrality of this particular sense of activity within activity recognition research can be explained by at least two factors. First, ADL monitoring is an important task in health care, and trained caregivers devote considerable time and effort to measure and track ADL accomplishment. Secondly, ADLs are general because they are elementary activities that people perform on a daily basis, and therefore interest goes beyond the health care realm.

In disregard of the variations of the activity concept, the representation and analysis of physical activities as computer entities rely altogether on the signals and data provided by activity sensing.

2.1.2. Activity sensing

A wide assortment of sensors form the sensing basis of HAR, and each type holds its own strengths and suitable applications. Sensors range from relatively simple instruments with discrete output (e.g. tilt-ball switches) and sensors with continuous output (e.g. inertial sensors: gyroscopes and accelerometers, covered extensively in [6]), to more elaborate sensing methods such as audio and video processing. While video cameras were once the most commonly deployed sensors for activity recognition, accelerometers have gained presence within the field as of recent. This change is supported by the nature of the physical quantity measured by these sensors, which is highly adapted to the body motions involved in physical activities.

In addition to the said devices, other specialized sensors are also employed. Techniques making use of time-of-flight cameras and depth sensors have emerged in the last few years, overcoming the sensitivity to scale and view in video-based approaches [9]. Fiber-optical sensors measure posture in [10]. Foam pressure sensors are able to transform muscle contractions into raw data [11]. Moreover, various kinds of highly specialized instruments such as oximetry sensors [12], skin conductivity sensors [13] and electrocardiographs [14], do appear in HAR studies.

Depending on the characteristics of the specific activity, recognition performance can be improved by using the same type of sensor at multiple body locations, assembling networks of heterogeneous sensors, or integrating a variety of them into a single device [15]. This multimodal sensing enables sensor fusion or merge by combining two or more complementary forms of data.

2.1.3. Activity recognition

In continuous development from the late 1990s, sensor-based activity recognition involves a series of tasks that can be summarized as follows: i) choose and deploy sensors (as per types described above); ii) collect, store and process sensor data; iii) generate computational data models; iv) develop or select algorithms to infer activities based on sensor data [5]. The second task demands a more technical account, so it is not covered here. As for the other three tasks they correspond to the sensing and modeling parts of HAR, and the next section introduces a classification of the associated techniques.

2.2. Activity recognition classification

Depending on the chosen aspect of HAR models and techniques can be segmented in different ways. None of these alternative classifications is to be taken as unique, since they are more

the product of a decision than something reflecting properties. Two non-exclusive criteria are outlined in this section: capture or sensing method, and model type. A general guide to these classifications is presented in table **2-1**, including advantages and disadvantages of each category.

Criterion	**Classes**	**Advantages**	**Disadvantages**
Capture method	Video-based	Very rich and easy to capture raw data	Complex modeling and privacy-related issues. Sensitivity to angle and depth
	Wearable-sensors	First-hand data capture and possibility of re-purposing mobile devices	Obtrusive and with limited autonomy. Privacy-related issues. DSP expertise advised for inertial sensors
	Smart spaces	Rich multi-modal data enabling sensor fusion and merge	Hard to reproduce and non-naturalistic sensing conditions
Model type	Generative	Non-deterministic probabilistic classification. Synthetic generation of datasets once the distribution is modeled	Extensive amounts of data required to perform well
	Discriminative	Similarity or rule-based reasoning. Easier to train with limited amounts of data	Usually paired with heavy feature-engineering. Focus on representing boundaries between classes and not classes themselves (in terms of features)

Table 2-1: Alternative classifications of HAR approaches

2.2.1. Activity recognition by capture method

This criterion separates activity recognition studies into video-based and non-image sensor-based. Video-based activity recognition depends on 2D and 3D video analysis techniques to extract human activity data. And non-image sensor-based activity recognition is further divided into research threads with wearable sensors and studies making use of smart spaces.

Video-based activity recognition

Visual or video recognition studies are often designed around four main steps: initialization of the system and the model, tracking, pose estimation, and recognition [16]. First step corresponds to instantiation of kinematic structure, shape and appearance of a subject. Tracking involves background subtraction, object segmentation, and discovery of temporal correspondences. And finally, pose estimation and recognition try to infer the configuration of the skeletal structure of a subject. This pipeline exploits video as a rich source of raw input data to recognize activities, but at the same time exposes a privacy issue regarding the common perception of cameras as recording devices ([5], [17]).

Wearable-sensing activity recognition

In relation to activity recognition with wearable sensors (sometimes called *attached sensors*), one of the most recognizable studies is *MSP*, presented in [15]. It went through many iterations until it ended up integrating seven sensors in a custom-designed portable device. This project evolved along four years, making adjustments not only to the sensor hardware, but also to the context model. As a result of this process it was reassured the superior discriminative capacity of inertial sensors and microphones, leaving behind other sensors with comparatively ambiguous reports.

But despite widespread adoption, activity monitoring by means of wearable sensors suffers from limitations. The majority of wearable sensors need to run continuously and impose a hands-free operation. This may present difficulties in everyday application scenarios, such as the willingness to use wearable sensors and the viability or ability to wear them. Technical issues include the size, ease of use, and autonomy (in terms of battery life). Some of these topics have been addressed by re-purposing mobile devices as wearable sensors, with studies going back as far as 2006 already deploying them [18].

Smart spaces activity recognition

As for studies requiring smart spaces some of them fit in the category of dense sensing, which refers to the practice of attaching sensors to items in order to monitor activities via user-object interactions. The approach is based on the fact that activities can be characterized by the objects that are manipulated during their performance. An example of this trend can be found in [19], a medical study aimed to evaluate consistency in activities of daily living.

A classification of activity recognition using capture method alone is often unreliable, with studies combining distinct kinds of sensors (e.g. [20] combines a three dimensional video-tracking system with wearable microphones to monitor inhabitants of an experimental setting). And in a similar way many authors defend a multi-modal or fusion perspective, com-

bining ambient and wearable sensors [16]. This leads to a second classification of activity recognition focused on the modeling itself [5]. According to this aspect HAR models can be classified as probabilistic or discriminative. The next section offers an overview of the current techniques in each group.

2.2.2. Activity recognition by model

This classification schema takes into account the models performing the actual recognition of the activities. In this sense the majority of approaches fall within the generative or discriminative sets.

Generative models

The generative set of HAR techniques includes models with support for temporal representation, which require enough data to fully learn the probabilistic representation they intend to cover. Hidden Markov Models (HMMs) are representative of this class.

HMMs are based on a series of hidden states, each one of them with a probability distribution. Every state accounts for an action or activity primitive, and state transitions (representing activities) follow a set of probabilities [6]. A so-called *observation*, to be compared with features extracted from sensors, can be generated for each state. Then activity recognition proceeds by training HMMs on some previously defined classes, in order to finally test those HMMs with new observations ([20], [21]).

HMMs are massively used in activity and context recognition projects, ranging from recognition of assembly tasks with microphones and accelerometers [22] and physical activity classification from on-body accelerometers [6], to multi-modal and multi-person analysis of human behavior in a smart home setting [20].

Other generative models taking part in HAR studies include Naïve Bayes as deployed in [23], and Dynamic Bayesian Networks (DBNs) as found in [24].

Discriminative models

Discriminative modeling goal is to set boundaries between classes instead of focusing on the representation (as probabilistic models do). In this sense it avoids the requirement of extensive training data affecting probabilistic studies.

The classification objective here means HAR tries to define or learn criteria to disaggregate activities into classes $A = \{A_1, A_2, ..., A_m\}$. This takes the form of a model M to predict a n-long activity sequence on the basis of an ordered set of –possibly multi-modal– sensor

readings $\boldsymbol{s} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_t, ..., \boldsymbol{d}_n\}$. Predicted activities ($\hat{A}$) and ground-truth activities (A^*) match equations 2-1 and 2-2, with a loss function ($\mathcal{L}$) measuring their relative distance in equation 2-3.

$$\hat{A} = \{\hat{A}_j\}_{j=1}^{n} = M(\boldsymbol{s}), \hat{A} \in A \tag{2-1}$$

$$\hat{A} = \{\hat{A}_j\}_{j=1}^{n} = M(\boldsymbol{s}), \hat{A} \in A \tag{2-2}$$

$$\mathcal{L} = (M(\boldsymbol{s}), A^*) \tag{2-3}$$

Usually the model function M acts on an intermediate function calculated over the sensor readings, so $M(\boldsymbol{s})$ becomes $M(\phi(\boldsymbol{s}))$, with ϕ denoting a feature projection function [25]. The specific nature of this mapping function and its relation to the raw data extends over the diversity of preprocessing and feature engineering.

Nearest neighbor model, where a current set of observations is compared against templates extracted from training data, represents a first and simple discriminative model for HAR. Another common discriminative technique successfully adopted from machine learning relies on support vector machines (SVMs). They are based on the automatic search of the data points closer to the boundaries, which then become the actual boundaries [5]. Reviewed works include studies where SVMs are deployed to detect body postures in a smart home environment [20], and to infer activities by means of the inertial sensors in smart phones [26].

2.3. Deep learning and activity recognition

The majority of the surveyed research works rely heavily on some kind of previous knowledge or feature engineering over the raw sensor data. Taken as a whole, conventional machine learning applied to pattern recognition demands the transformation of the raw data into features, that is into a representation suitable for the learning model. Deep learning closes this gap with its representation learning capabilities [27]. At the core, deep learning replaces traditional machine learning assumptions (e.g. the local constancy prior) with the general idea that the analyzed data was generated at one or more hierarchical levels by a *composition of factors*. This change in strategy translates into an increased capability to cope with the *curse of dimensionality* affecting the scalability of previous efforts [28].

In general, each stacked layer in a deep learning architecture performs a non-linear transformation on the output of the previous layer, extracting a hierarchy from high-level to low-level features. Commonly deployed deep learning models incorporating this layout can be categorized into unsupervised, supervised, and hybrid approaches [1]. The latter class encompasses networks discriminative in outcome but assisted with generative or unsupervised

models in their optimization or regularization stages, and generative networks where parameters are estimated by supervised means. Table **2-2** provides a list of related models within each class of this categorization. Sections below expand on some architectures of interest to HAR research.

Network class	Representative models/architectures
Unsupervised	**Generative:** Restricted Boltzmann Machine (RBM) Deep Boltzmann Machine (DBM) Deep Belief Network (DBN) Denoising Autoencoder **Discriminative:** Deep Autoencoder Sparse Coding Network
Supervised	**Discriminative:** Deep-structured Conditional Random Field (CRF) Deep Neural Network (DNN) Deep Stacking Network (DSN) Recurrent Neural Network (RNN) Convolutional Neural Network (CNN) Time-delay Neural Network (TDNN) Hierarchical Temporal Memory (HTM)
Hybrid	**Generative:** DBN-initialized supervised DNN DBN-initialized DNN, further tuned with CRF DBN pre-trained CNN

Table 2-2: General classification of deep learning approaches and representative models as proposed in [1]

2.3.1. Deep learning models for activity recognition

HAR research is increasingly embracing the advantages of deep learning models, replacing heuristic feature-extraction with their high-level abstraction capabilities. Deep generative models as those listed in table **2-2** further empower HAR to perform well under unsupervised learning conditions, including online classifying tasks [25]. In this section three deep models commonly deployed for activity recognition tasks are briefly covered.

Restricted Boltzmann Machines (RBMs)

RBMs are artificial neural networks with a hidden layer and a visible layer, where units in each group are not interconnected and hidden units establish undirected symmetrical connections with visible units [29]. The stacked version of the RBM is the Deep Belief Network

(DBN), which have been adopted alongside pre-training in activity recognition models.

One early application of RBMs to HAR [30] implements a DBN with two hidden layers of 1,024 units each, with the goal of learning "universal features". The performance reports for this model compare it to a Nearest Neighbor classifier with statistical features and to a PCA model, outperforming both with accuracies between 74 % and 90 % on four public datasets.

In [31] a DBN composed of two RBMs containing from 512 to 2,048 units undergoes pre-training on unlabeled data from wrist-worn accelerometers, in order to learn a generative model for 34 patients diagnosed with Parkinson's Disease (PD). This tuned model is then trained with self-labeled data from the same group of patients. The study sought to close the gap between the expert movement analysis performed in laboratory settings and the self-reporting of the patients at home. However, the poorly labeled samples translated into low f1 scores under 0.6 (despite medical expertise being applied to design the 91 features fed into the model).

A very successful context and behavior analysis in a mobile context is portrayed by the *Deepear* audio analysis system [32]. Four coupled DNNs of stacked RBMs work together in this architecture with roles of ambient audio scene analysis, speaker identification, emotion recognition, and stress detection. The main contribution here is substituting the Gaussian Mixture Model (GMM) in the audio processing pipeline with a deep learning model.

Recurrent Neural Networks (RNNs)

RNNs are particularly well adapted to process the temporal sequences of inputs in HAR because their units form a directed graph. Indeed, long short-term (LSTM) units –a special kind of building units for RNN– are considered to be highly effective in learning the temporal dependencies found in human motion. LSTM memory-enabled networks can be combined with the feature learning capabilities of convolutional layers for multi-modal HAR [33].

An advanced application of RNNs to mobile device authentication based on motion patterns is introduced in [34]. Temporal feature extraction via an extension of Clockwork RNN (capable of modeling multiple time scales) serves as input to a classification stage with Gaussian Mixture Model. This sophisticated Dense Clockwork extension to RNN addresses the limitations of the original model when coping with similar data at different points in time (shift-variance).

Convolutional Neural Networks (CNNs)

ConvNets are designed to process data that comes in the form of multiple arrays, while taking advantage of four key properties of raw input signals: local connections, shared weights,

pooling, and stacking of multiples layers [28]. For this purpose common architectures of convolutional networks consist on a series of stages, with the first few combining convolutional and pooling steps.

Units in a convolutional layer are grouped in feature maps, and each unit in them is connected to local patches in the feature maps of the prior layer through a *filter* (set of weights). The result of this local weighted sum is then fed to a non-linear function (e.g. ReLU). Local sharing of the set of weights takes place for units in the same feature map, but this schema does not extend to different feature maps within a layer [35].

In its turn the role of the pooling layer is merging semantically similar features. This way, a typical pooling unit calculates either average or maximum of a local patch of units in a feature map (or in a few feature maps), reducing the dimension of representation and introducing invariance to small shifts and distortions [27].

As for works making use of CNNs for HAR in [36] a standard architecture of convolutional and pooling layers followed by a fully connected step learns a representation for the set of six activities in Smartlab's dataset (the same experimental dataset to be used throughout this work). After "greedy-wise" hyperparameter tuning the best performing network features a wide receptive field and a small pooling size over three convolutional and pooling steps.

A HAR-targeted extension of the basic convolutional and pooling layers is presented in [37], where corresponding filters are applied along the temporal dimension for each sensor, and the resulting activation maps form a unified input for the CNN.

2.4. Activity recognition datasets

Until recently there was limited availability of public datasets to evaluate activity recognition models. HAR studies tended to generate their own limited-scope datasets, not making them available for repeatability and comparison of results. Over the last few years more and more universities and research groups have been releasing HAR datasets, effectively supporting an incremental improvement in the quality of works within the field. A summary of the datasets presented in this section is provided in table **2-3**.

From MIT's PlaceLab experimental installations comes one of the first datasets ever released to the general public, comprising a set of ten everyday household activities (e.g. preparing a recipe, doing the dishes, making a bed, talking on the phone, among others). These were performed during a period of four hours by a single subject wearing a heart rate monitor and two accelerometers attached to his limbs and hip [38]. This dataset and a few additional ones from the same experimental space were published in PlaceLab's page, but at the time

of writing this document they are no longer available for download.

The Physical Activity Monitoring dataset, *PAMAP2* [39], also articulates inertial measures alongside heart rates. It contains samples from 9 subjects and 18 activities captured in a controlled setting, with a protocol specifying required and optional activities. Some of the most frequent activity classes in HAR works are covered here (e.g. walking, standing, running, walking upstairs, walking downstairs) as well as other less studied (e.g. jumping rope and playing soccer).

With the aid of similar inertial units the University of California at Berkeley gathered signals on a sample of 20 subjects [40]. This Wearable Action Recognition Dataset (*WARD*) includes 13 activities of daily living and is distributed as unprocessed raw signals, with some data missing for technical failures of the motion devices.

Expanding on the body-worn sensors prevalent in the HAR datasets just mentioned, a multimodal activity dataset from Carnegie Mellon combines them with video, audio, RFID, motion capture, and physiological sensors in a smart setting [41]. The captured activity data from the 43 subjects in this Multi-Modal Activity Dataset (*CMU-MMAC*) corresponds to cooking-related tasks for a fixed set of five recipes.

Another dataset making use of a smart space was released in 2010 by the European research project Opportunity, and holds samples from 4 subjects interacting with wearable, object, and ambient sensors [42]. Labeled data from 72 sensors unveils a complete activity hierarchy: from modes of locomotion and low-level object-related actions, to mid-level gesture classes, and 5 *very* high-level activity classes (e.g. preparing a sandwich). In this case a top level script provided to subjects was expected to allow for naturalistic expressions of performed activities.

The University of Southern California released the *USC-HAD* dataset in 2012, which includes gyroscope signals from 12 scripted activities carried out by 14 subjects wearing an off-the-shelf sensing device (MotionNode) on the front right hip [43]. Since this is a wired device a laptop was being carried by the subjects during their scripted performance. As a remark, for this dataset there was an explicit interest in selecting a diverse group of individuals as experimental subjects, to account for more of the variability in motion patterns of the wider population.

Other datasets rely on smart phones as sensing devices for practical purposes and aiming to exploit the pervasive aspect of HAR. Representative of this class is Smartlab's HAR Dataset, available at UCI Irvine Machine Learning Repository [44] under two revisions ([2], [45]). The original release comprises 6 basic activities (walking, walking upstairs, walking downstairs,

lying down, sitting and standing up) and the updated revision adds an equally sized set of intermediate "postural transitions". Experimental setup consists of 30 subjects that wore the smart phone on their waists while performing the scripted activities. Additional details of this dataset are presented in the next chapters, as this is the dataset chosen for the experimental sections of this work.

WISDM dataset contains set of 6 activities closely matching Smartlab's –substituting *laying down* with *jogging*– and also deploys smart phones for motion signal capture from 29 subjects [46]. However, in this case only accelerometer signals are collected and published.

Dataset	**Year**	**Subjects**	**Activities**	**Sensors**	**Ref.**
PlaceLab	2005	1	10 ADLs	(1) Heart rate monitor (3) Body-worn accelerometers	[38]
WARD	2008	20	13 ADLs	(5) Body-worn inertial units with accelerometer and gyroscope	[40]
CMU-MMAC	2008	43	5 food preparing activities	(6) Video cameras (5) Microphones (16) IR cameras (9) Inertial units (1) RFID bracelet	[41]
Opportunity	2010	4	16 activities in a hierarchy	(7) Body-worn inertial units (12) Object sensors (21) Ambient sensors	[42]
USC-HAD	2012	14	12 ADLs	(1) Body-worn inertial unit with accelerometer and gyroscope	[43]
PAMAP2	2012	9	18 ADLs	(3) Body-worn inertial units (1) Heart rate monitor	[39]
Smartlab's HAR (V. 1.0)	2012	31	6 ADLs	(1) Smart phone embedded accelerometer and gyroscope	[2]
WISDM	2012	29	6 ADLs	(1) Smart phone embedded accelerometer and gyroscope	[46]
Smartlab's HAR (V. 2.1)	2014	31	6 ADLs plus 6 transient activities	(1) Smart phone embedded accelerometer and gyroscope	[45]

Table 2-3: Publicly available HAR datasets

3 Smartlab's HAR dataset exploration

Smartlab's HAR dataset is one in a growing collection of publicly available datasets to test and benchmark models in this domain of knowledge. Its defining characteristic is the repurposing of mid-range smart phones with embedded accelerometers and gyroscopes as sensor hubs, opening the possibility of extending or comparing its raw data with new signals, to be gathered with similar devices. Also, it retains great value for newcomers to the activity recognition field as one of the first well documented datasets for motion-related inertial signals.

The sections in this chapter describe the contents of the dataset, and register an attempt to replicate the preprocessing and feature extraction stages performed by its proponents [2]. Additionally, by using guidelines in their works, an SVM learning model applied to the original raw data tries to match the reported scores. This will serve as a baseline for the next chapters, which propose an alternative learning model that builds upon feature discovery and convolutional neural networks.

3.1. Dataset overview

There are two available versions of Smartlab's dataset in the UCI Machine Learning Repository, released with more than two years in between[1]. First version (1.0) has a donation date of December 2012, while the updated revision (2.1) was added to the repository by the mid of 2015. Both share a one-time experimental setup of 30 subjects aged between 19 and 48 years, wearing waist-mounted Samsung Galaxy S2 phones while performing a scripted sequence of six activities. Complete video footage of the experiments allowed a posterior tagging of the set of activities evaluated.

3.1.1. Dataset revisions and associated models

Version 1.0 of Smartlab's dataset includes already windowed acceleration and velocity signals alongside a full collection of the 561 features calculated for each activity window. Samples are labeled with identifiers for activity, subject and experiment number. No other demographic information like gender or age is provided regarding the participants. In the corresponding paper [2] this dataset comes paired with an SVM model for recognition of three "non-dynamic"

[1] At the time of writing the two versions of Smartlab's HAR dataset are available for download in [47] and [48], respectively.

and three "dynamic" activities, as listed in table **3-1**. This model is depicted in figure **3-1**.

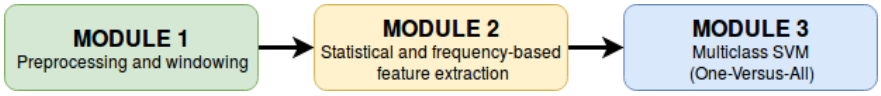

Figure 3-1: First SVM model (offline) proposed by authors of Smartlab's HAR dataset

Activity	Abbreviation	Group
Walking	WK	Dynamic (dyn)
Walking downstairs	WD	Dynamic (dyn)
Walking upstairs	WU	Dynamic (dyn)
Standing	SU	Non-dynamic (ndyn)
Sitting	SD	Non-dynamic (ndyn)
Laying down	LD	Non-dynamic (ndyn)

Table 3-1: Set of activities in version 1.0 of Smartlab's HAR dataset (includes abbreviations to be used throughout this text)

The more recent revision of the dataset (2.1) introduces the raw and non-windowed inertial sensor data from the experimental sessions and adds six new *postural transition* classes [2]. These were intended to support the performance of a more sophisticated heuristic-augmented SVM activity recognition model for the non-dynamic activities (which showed consistently lower scores under the original SVM model). The extended set of activities is shown in table **3-2** and the figure **3-2** presents the overall model related to this version of the dataset. As a remark, this model is designed to be an online architecture, enabling real-time classification as demonstrated in [49].

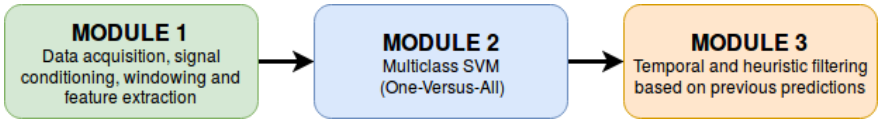

Figure 3-2: Second SVM model (online) proposed by authors of Smartlab's HAR dataset

In what follows the raw signals as well as the preprocessing stage shared by the two iterations of the dataset are examined, as a preamble to the experimentation sections of this chapter, where two baseline models are built to process the features and raw data in Smartlab's dataset.

[2]The updated version also renames the former activity groups as "static postures" (for the non-dynamic subset) and "ambulation activities" (previously dynamic activities)

Activity	Group
Walking	Ambulation activity
Walking downstairs	Ambulation activity
Walking upstairs	Ambulation activity
Standing	Static posture
Sitting	Static posture
Laying down	Static posture
Stand-to-sit	Postural transition
Sit-to-stand	Postural transition
Sit-to-lie	Postural transition
Lie-to-sit	Postural transition
Stand-to-lie	Postural transition
Lie-to-stand	Postural transition

Table 3-2: Extended set of activities in version 2.1 of Smartlab's HAR dataset (including relabeling of previous activity groups)

3.1.2. Dataset raw signals

The raw form of signals in the later revision of Smartlab's dataset is composed of real valued time series for the linear acceleration ($\boldsymbol{a}$) and angular velocity ($\boldsymbol{w}$), captured at a sampling rate of 50Hz by the embedded accelerometer and gyroscope in the mobile device. Table **3-3** shows the actual ranges for the attributes of over 1,150,000 raw values included in the dataset, whereas the figure **3-3** is a sample plot of 128 contiguous data points (an activity window) for the walking upstairs activity.

Attribute	Description
a_x,a_y,a_z	Real value in range [-2.0,2.0]. Linear acceleration over each axis expressed in meters per second, with 1/50th of a second resolution.
w_x,w_y,w_z	Real value in range [-7.0,7.0]. Instantaneous angular velocity over each axis expressed in radians per second, with 1/50th of a second resolution

Table 3-3: Smartlab's HAR dataset attributes

Raw signals in version 2.1 are distributed in 122 plain text files, half of which contain accelerometer signals and the other half are angular velocities from the gyroscope. This set comprehends a total of 61 experiments with 30 participants. And finally, a labels file associates ranges of readings in the first pair of files to the activity -or postural transition- identifier.

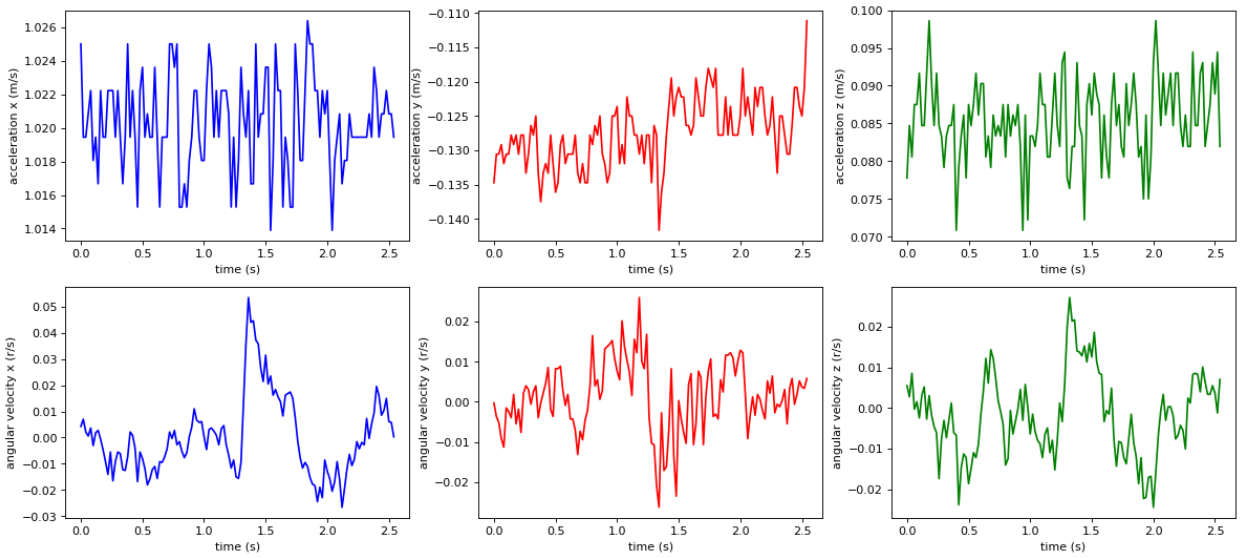

Figure 3-3: Plot of the triaxial signals in a sample activity window from Smartlab's HAR dataset (walking upstairs activity)

3.1.3. Reference model preprocessing

From [26] the sequence of seven preprocessing steps in table **3-4** is identified in order to obtain the signals for the feature extraction stage. This preprocessing setup, reused throughout the models proposed by authors of Smartlab's HAR dataset, yields a total of 17 time and frequency signals out from the original $\boldsymbol{a}$ and $\boldsymbol{w}$, as it is summarized in table **3-5** and **3-6**. Please note the change in nomenclature from vector notation to more descriptive signal names prefixed with t (time-domain) or f (frequency-domain).

Step	Transformation	Result
1	Apply median filter to raw signals $\boldsymbol{a}$ and $\boldsymbol{w}$, with $\boldsymbol{a} = \{a_x, a_y, a_z\} \in R^3$ and $\boldsymbol{w} = \{w_x, w_y, w_z\} \in R^3$	Signals $\boldsymbol{a}_{NR1}$ and $\boldsymbol{w}_{NR1}$ with first step of noise reduction applied
2	Apply third-order low pass Butterworth filter to $\boldsymbol{a}_{NR1}$ and $\boldsymbol{w}_{NR1}$ with 20 Hz as constant corner frequency	Signals $\boldsymbol{a}_{NR2}$ and $\boldsymbol{w}_{NR2}$ with second step of noise reduction applied
3	Sample time signals $\boldsymbol{a}_{NR2}$ and $\boldsymbol{w}_{NR2}$ in fixed-width sliding windows of 2.56 seconds length and 50 % overlap	Data segmentation of signals $\boldsymbol{a}_{NR2}$ and $\boldsymbol{w}_{NR2}$ (from this step onwards referred as $\boldsymbol{tBodyGyro}_{xyz}$ signal)

4	Apply third-order low pass Butterworth filter to windowed $\boldsymbol{a}_{NR2}$ with 0.3 Hz as constant corner frequency	Body and gravity acceleration ($\boldsymbol{tBodyAcc}_{xyz}$ and $\boldsymbol{tGravityAcc}_{xyz}$)
5	Derive in time body linear acceleration $\boldsymbol{tBodyAcc}_{xyz}$ and angular velocity $\boldsymbol{tBodyGyro}_{xyz}$	Jerk signals ($\boldsymbol{tBodyAccJerk}_{xyz}$ and $\boldsymbol{tBodyGyroJerk}_{xyz}$)
6	Use euclidean norm over body linear acceleration, gravity acceleration, angular velocity, body acceleration jerk, and body angular velocity jerk	Magnitudes of three-dimensional signals: $tBodyAccMag$, $tGravityAccMag$, $tBodyAccJerkMag$, $tBodyGyroMag$, $tBodyGyroJerkMag$
7	Apply FFT to $\boldsymbol{tBodyAcc}_{xyz}$, $\boldsymbol{tBodyAccJerk}_{xyz}$, $\boldsymbol{tBodyGyroJerk}_{xyz}$, $tBodyAccJerkMag$, $tBodyGyroMag$, $tBodyGyroJerkMag$	Frequency-domain signals: $\boldsymbol{fBodyAcc}_{xyz}$, $\boldsymbol{fBodyAccJerk}_{xyz}$, $\boldsymbol{fBodyGyro}_{xyz}$, $fBodyAccJerkMag$, $fBodyGyroMag$, $fBodyGyroJerkMag$

Table 3-4: Preprocessing steps for Smartlab's HAR dataset

Time-domain signal	**Description**
$\boldsymbol{tBodyAcc}_{xyz}$	Body acceleration
$\boldsymbol{tGravityAcc}_{xyz}$	Gravity acceleration
$\boldsymbol{tBodyAccJerk}_{xyz}$	Body acceleration jerk
$\boldsymbol{tBodyGyro}_{xyz}$	Body angular speed
$\boldsymbol{tBodyGyroJerk}_{xyz}$	Body angular speed jerk
$tBodyAccMag$	Magnitude of body acceleration
$tGravityAccMag$	Magnitude of gravity acceleration
$tBodyAccJerkMag$	Magnitude of body acceleration jerk
$tBodyGyroMag$	Magnitude of body angular speed
$tBodyGyroJerkMag$	Magnitude of body angular speed jerk

Table 3-5: Time signals out from the preprocessing stage of Smartlab's HAR dataset

Frequency-domain signal	Description
$\boldsymbol{fBodyAcc}_{xyz}$	Body acceleration
$\boldsymbol{fBodyAccJerk}_{xyz}$	Body acceleration jerk
$\boldsymbol{fBodyGyro}_{xyz}$	Body angular velocity
$\boldsymbol{fBodyAccMag}$	Magnitude of body acceleration
$fBodyAccJerkMag$	Body acceleration jerk magnitude
$fBodyGyroMag$	Magnitude of body angular speed
$fBodyGyroJerkMag$	Magnitude of body angular speed jerk

Table 3-6: Frequency signals out from the preprocessing stage of Smartlab's HAR dataset

3.1.4. Reference model calculated features

561 features are extracted from the windowed time and frequency signals in the preprocessing stage by applying the set of functions of table **3-5** and table **3-6**. Dataset authors explain their selection of features by referring previous works of their own ([2], [50]) as well as external ([51], [52], [53]). From those sources the authors conclude that frequency-domain measures and calculations over gyroscope signals do not offer considerable improvement, but they decide to keep the associated features only for informational and discriminative purposes.

Function	Description
mean()	Mean value
std()	Standard deviation
mad()	Median absolute deviation
max()	Largest value in the array
min()	Smallest value in the array
sma()	Signal magnitude area
energy()	Sum of squares divided by number of values
iqr()	Interquartile range
entropy()	Signal entropy
arCoeff()	Autoregression coefficients (with Burg order 4)
correlation()	Correlation coefficient between two signals
maxFreqIdx()	Index of the frequency component with the largest magnitude
meanFreq()	Weighted average of the frequency components to obtain a mean frequency
skewness()	Skewness of the frequency-domain signal
kurtosis()	Kurtosis of the frequency-domain signal
bandsEnergy()	Energy of a frequency interval in the 64 bins of the FFT for each window
angle()	Angle between two vectors

Table 3-7: Feature calculations in reference model for Smartlab's dataset (adapted from [2])

3.1.5. Reference model process and results

Out of the two models in figure **3-1** and figure **3-2** the simpler SVM accompanying the first version of Smartlab's dataset is chosen as benchmark, because it is the better documented one (making reproduction of architecture and results more approachable). Overall procedure by the authors in deploying this machine learning model can be summarized as follows: *i)* preprocessed dataset is randomly partitioned with 70/30 ratio between training and testing sets, taking each subject's samples as whole units over which partitioning proceeds; *ii)* an SVM binary classifier with Gaussian kernel is selected as model, extending it to the multi-class case by means of One-Versus-All (OVA) technique; *iii)* SVM hyperparameters are selected through 10-fold cross-validation. Table **3-8** shows the reference results from this model, after being trained and tested with version 1.0 of the dataset.

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**492**	1	3	0	0	0	99 %
Walking Upstairs	18	**452**	2	0	0	0	96 %
Walking Downstairs	4	6	**410**	0	0	0	98 %
Sitting	0	2	0	**432**	57	0	88 %
Standing	0	0	0	14	**518**	0	97 %
Laying down	0	0	0	0	0	**537**	100 %
Precision	96 %	98 %	99 %	97 %	90 %	100 %	**96 %**

Table 3-8: Confusion matrix of classification results for reference model of Smartlab's HAR dataset (reproduced from [2])

3.2. Dataset raw signals exploratory analysis

Partial preprocessing of the signals was executed in a preliminary exploration of the dataset, following the guidelines provided in [26]. It started with each component of linear acceleration and angular velocity signals being fed through a median filter with window size 3, to smooth out peaks and random noise typical of *MEMS* sensor technology [54]. Afterwards, the already modified signals were aggregated by means of a sliding window, matching the length choice made in the dataset introductory paper (i.e. 2.56 seconds or 128 samples). A low-pass filter was finally applied to filter out the gravitational component while keeping body motion readings.

An statistical analysis was then performed over the inertial signals in order to obtain a better understanding of the raw data and its structure. Four measures (mean, maximum, minimum, and standard deviation) were calculated over each body motion signal component at window

scale. Two sets of patterns are identified in the data:

- *Non-dynamic activities specifics:* the best separation between so-called static and dynamic activities is achieved by combining the readings from accelerometer and gyroscope. In particular, when comparing x-axis maximum angular velocities a characterization of static activities emerges based on their low angular velocities. This pattern appears in figure **3-4**.
- *Dynamic activities specifics:* as per expectation accelerations matching the axis of horizontal subject movement for walking upstairs and walking downstairs activities show a similar distribution of minimum and maximum values. Also, within dynamic activities there is a high degree of dispersion for linear x-axis acceleration data points, which is consistent with the vertical movement of subjects during stairway-related motions. However, oddly enough a horizontal displacement like the one performed for walking activity shows a comparatively high standard deviation around 0.4 meters per second. Figure **3-5** shows these traits of dynamic activities.

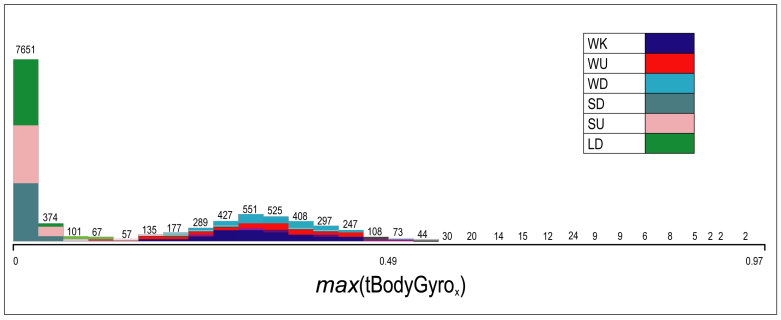

Figure 3-4: Histogram for maximum of $tBodyGyro_x$ clustering together gyroscope signals for non-dynamic activities (SD, SU and LD from table **3-1**)

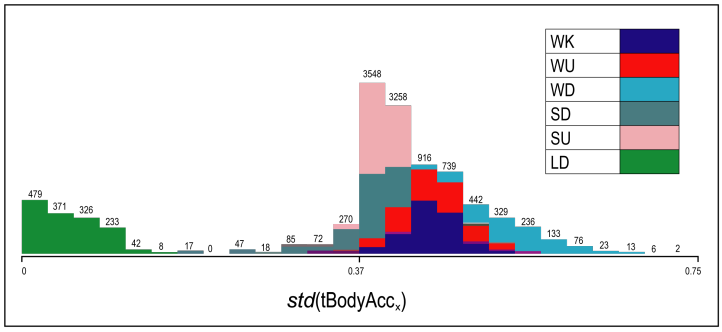

Figure 3-5: Histogram for standard deviation of $tBodyAcc_x$ showing high dispersion for dynamic activities (WK, WU and WD from table **3-1**)

3.3. Baseline model with feature input

The first effort to replicate the results from the original SVM model attached to version 1.0 of Smartlab's HAR dataset takes as input the full collection of windowed features (supplied with this release). To this end, it deploys an SVM with Gaussian kernel as described in the reference works [2], evaluating it against the 70/30 pre-segmented training and testing sets available for the dataset.

3.3.1. Experimental setup

No hyperparameter values are explicitly mentioned by the authors, so a 10-fold cross validation random search optimization takes place [55]. The input distributions and the resulting values for C (soft-margin constant) and γ (gamma free parameter) are shown in table **3-9**, where objective metric for optimization is accuracy. The selected values satisfy the general guidelines of a large C to penalize margin errors and a small γ to counter over-fitting the samples.

Hyper-parameter	**Input sampling distribution for random search**	**Selected value**
C	$f(x) = exp(-x)$ with 10^6 scale	$3{,}1622776601683795x10^6$
γ	$f(x) = exp(-x)$ with 10^{-3} scale	$5{,}623413251903491x10^{-3}$

Table 3-9: Random search parameters and results for hyperparameter optimization of baseline SVM model with feature input

3.3.2. Results

Experimental results for this baseline Gaussian kernel SVM with optimized parameters are shown in table **3-10**.

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**481**	7	8	0	0	0	97.0 %
Walking Upstairs	44	**417**	10	0	0	0	88.5 %
Walking Downstairs	10	10	**400**	0	0	0	95.2 %
Sitting	0	2	0	**457**	49	0	90.0 %
Standing	0	0	1	22	**533**	0	95.9 %
Laying down	0	0	0	0	0	**545**	100.0 %
Precision	90.0 %	95.6 %	95.5 %	95.4 %	91.6 %	100.0 %	**94.7 %**

Table 3-10: Confusion matrix of classification results for baseline Gaussian SVM model with feature input

As in the reference confusion matrix of table **3-8** the largest classification overlaps are related to walking samples wrongly labeled as walking upstairs and sitting samples misclassified as standing. The later pair of activities are considered non-dynamic by the authors, and this overlap led them to release an extended revision of the model. Remarkably, in an additional test a less complex linear kernel SVM with *C=1.0* got closer to the precision and recall scores achieved in the original research paper, as it is presented in table **3-11**.

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**492**	1	3	0	0	0	99.2 %
Walking Upstairs	18	**451**	2	0	0	0	95.7 %
Walking Downstairs	4	6	**410**	0	0	0	97.6 %
Sitting	0	2	0	**435**	54	0	88.6 %
Standing	0	0	0	16	**516**	0	97.0 %
Laying down	0	0	0	0	0	**537**	100.0 %
Precision	95.7 %	98.0 %	98.8 %	96.4 %	90.6 %	100.0 %	**95.5 %**

Table 3-11: Confusion matrix of classification results for baseline linear SVM model with feature input

3.4. Baseline model with raw signals input

After achieving similar scores to the reference model by means of a comparable SVM architecture the experimentation went on imitating the preprocessing and feature engineering stages in table **3-4**.

3.4.1. Experimental setup

To fulfill the steps listed in table **3-4** some assumptions had to be made, since there was no complete information regarding the preprocessing stage in the available dataset documentation: *i)* median filter size is 3; *ii)* third-order Butterworth filter is digital; *iii)* no ranking over the data is performed prior to interquartile range calculation; *iv)* windows of less than 128 samples at the end of per-subject experiment and activity chunks of raw data are kept as valid ones; *v)* features are normalized and bounded within $[-1, 1]$ (as described in readme file from version 1.0), but it is assumed these calculations are applied per set (train and test) rather than window-wise.

Table **3-12** references all the features successfully computed by applying functions in table **3-6** to the raw data. These functions were used only over the time signals in table **3-5**, for

a total of 134 extracted features.

A control score was established to compare these preprocessing and feature calculation steps with the reference pipeline. This was achieved by sampling the same 134 features from the train and test sets included in the Smartlab's dataset. Then a training and testing cycle was executed with these reference inputs by means of the best performing SVM model from the previous section. A second scenario tries to validate the calculated features by selecting the same test and train subjects of the previous setup but performing the preprocessing and feature extraction (instead of reusing the pre-computed features in version 1.0 of Smartlab's dataset). The same SVM model is evaluated with these new training and testing sets.

Preprocessing stage signal	**Calculated features**
$tBodyAcc_{xyz}$	$sma(tBodyAcc_{xyz})$
$tBodyAcc_x$	$mean(tBodyAcc_x)$, $std(tBodyAcc_x)$, $mad(tBodyAcc_x)$, $max(tBodyAcc_x)$, $min(tBodyAcc_x)$, $energy(tBodyAcc_x)$, $iqr(tBodyAcc_x)$
$tBodyAcc_y$	$mean(tBodyAcc_y)$, $std(tBodyAcc_y)$, $mad(tBodyAcc_y)$, $max(tBodyAcc_y)$, $min(tBodyAcc_y$, $energy(tBodyAcc_y)$, $iqr(tBodyAcc_y)$
$tBodyAcc_z$	$mean(tBodyAcc_z)$, $std(tBodyAcc_z)$, $mad(tBodyAcc_z)$, $max(tBodyAcc_z)$, $min(tBodyAcc_z)$, $energy(tBodyAcc_z)$, $iqr(tBodyAcc_z)$
$tGravityAcc_{xyz}$	$sma(tGravityAcc_{xyz})$
$tGravityAcc_x$	$mean(tGravityAcc_x)$, $std(tGravityAcc_x)$, $mad(tGravityAcc_x)$, $max(tGravityAcc_x)$, $min(tGravityAcc_x)$, $energy(tGravityAcc_x)$, $iqr(tGravityAcc_x)$
$tGravityAcc_y$	$mean(tGravityAcc_y)$, $std(tGravityAcc_y)$, $mad(tGravityAcc_y)$, $max(tGravityAcc_y)$, $min(tGravityAcc_y)$, $energy(tGravityAcc_y)$, $iqr(tGravityAcc_y)$
$tGravityAcc_z$	$mean(tGravityAcc_z)$, $std(tGravityAcc_z)$, $mad(tGravityAcc_z)$, $max(tGravityAcc_z)$, $min(tGravityAcc_z)$, $energy(tGravityAcc_z)$, $iqr(tGravityAcc_z)$
$tBodyAccJerk_{xyz}$	$sma(tBodyAccJerk_{xyz})$
$tBodyAccJerk_x$	$mean(tBodyAccJerk_x)$, $std(tBodyAccJerk_x)$, $mad(tBodyAccJerk_x)$, $max(tBodyAccJerk_x)$, $min(tBodyAccJerk_x)$, $energy(tBodyAccJerk_x)$, $iqr(tBodyAccJerk_x)$

$tBodyAccJerk_y$	$mean(tBodyAccJerk_y)$, $std(tBodyAccJerk_y)$, $mad(tBodyAccJerk_y)$, $max(tBodyAccJerk_y)$, $min(tBodyAccJerk_y)$, $energy(tBodyAccJerk_y)$, $iqr(tBodyAccJerk_y)$
$tBodyAccJerk_z$	$mean(tBodyAccJerk_z)$, $std(tBodyAccJerk_z)$, $mad(tBodyAccJerk_z)$, $max(tBodyAccJerk_z)$, $min(tBodyAccJerk_z)$, $energy(tBodyAccJerk_z)$, $iqr(tBodyAccJerk_z)$
$tBodyGyro_{xyz}$	$sma(tBodyGyro_{xyz})$
$tBodyGyro_x$	$mean(tBodyGyro_x)$, $std(tBodyGyro_x)$, $mad(tBodyGyro_x)$, $max(tBodyGyro_x)$, $min(tBodyGyro_x)$, $energy(tBodyGyro_x)$, $iqr(tBodyGyro_x)$
$tBodyGyro_y$	$mean(tBodyGyro_y)$, $std(tBodyGyro_y)$, $mad(tBodyGyro_y)$, $max(tBodyGyro_y)$, $min(tBodyGyro_y)$, $energy(tBodyGyro_y)$, $iqr(tBodyGyro_y)$
$tBodyGyro_z$	$mean(tBodyGyro_z)$, $std(tBodyGyro_z)$, $mad(tBodyGyro_z)$, $max(tBodyGyro_z)$, $min(tBodyGyro_z)$, $energy(tBodyGyro_z)$, $iqr(tBodyGyro_z)$
$tBodyGyroJerk_{xyz}$	$sma(tBodyGyroJerk_{xyz})$
$tBodyGyroJerk_x$	$mean(tBodyGyroJerk_x)$, $std(tBodyGyroJerk_x)$, $mad(tBodyGyroJerk_x)$, $max(tBodyGyroJerk_x)$, $min(tBodyGyroJerk_x)$, $energy(tBodyGyroJerk_x)$, $iqr(tBodyGyroJerk_x)$
$tBodyGyroJerk_y$	$mean(tBodyGyroJerk_y)$, $std(tBodyGyroJerk_y)$, $mad(tBodyGyroJerk_y)$, $max(tBodyGyroJerk_y)$, $min(tBodyGyroJerk_y)$, $energy(tBodyGyroJerk_y)$, $iqr(tBodyGyroJerk_y)$
$tBodyGyroJerk_z$	$mean(tBodyGyroJerk_z)$, $std(tBodyGyroJerk_z)$, $mad(tBodyGyroJerk_z)$, $max(tBodyGyroJerk_z)$, $min(tBodyGyroJerk_z)$, $energy(tBodyGyroJerk_z)$, $iqr(tBodyGyroJerk_z)$
$tBodyAccMag$	$mean(tBodyAccMag)$, $std(tBodyAccMag)$, $mad(tBodyAccMag)$, $max(tBodyAccMag)$, $min(tBodyAccMag)$, $sma(tBodyAccMag)$, $energy(tBodyAccMag)$, $iqr(tBodyAccMag)$
$tGravityAccMag$	$mean(tGravityAccMag)$, $std(tGravityAccMag)$, $mad(tGravityAccMag)$, $max(tGravityAccMag)$, $min(tGravityAccMag)$, $sma(tGravityAccMag)$, $energy(tGravityAccMag)$, $iqr(tGravityAccMag)$

$tBodyAccJerkMag$	$mean(tBodyAccJerkMag)$, $std(tBodyAccJerkMag)$, $mad(tBodyAccJerkMag)$, $max(tBodyAccJerkMag)$, $min(tBodyAccJerkMag)$, $sma(tBodyAccJerkMag)$, $energy(tBodyAccJerkMag)$, $iqr(tBodyAccJerkMag)$
$tBodyGyroMag$	$mean(tBodyGyroMag)$, $std(tBodyGyroMag)$, $mad(tBodyGyroMag)$, $max(tBodyGyroMag)$, $min(tBodyGyroMag)$, $sma(tBodyGyroMag)$, $energy(tBodyGyroMag)$, $iqr(tBodyGyroMag)$
$tBodyGyroJerkMag$	$mean(tBodyGyroJerkMag)$, $std(tBodyGyroJerkMag)$, $mad(tBodyGyroJerkMag)$, $max(tBodyGyroJerkMag)$, $min(tBodyGyroJerkMag)$, $sma(tBodyGyroJerkMag)$, $energy(tBodyGyroJerkMag)$, $iqr(tBodyGyroJerkMag)$

Table 3-12: Features extracted for baseline SVM model with raw signals input

3.4.2. Results

Overall scores shown in **3-13** for the control scenario mentioned above are slightly affected by the selection of a subset of features, but nonetheless the prediction performance ranks very high (as measured by overall accuracy).

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**481**	7	8	0	0	0	97.0 %
Walking Upstairs	45	**415**	11	0	0	0	88.1 %
Walking Downstairs	12	11	**397**	0	0	0	94.5 %
Sitting	0	2	0	**453**	53	0	89.2 %
Standing	0	0	1	22	**533**	0	95.9 %
Laying down	0	0	0	0	0	**545**	100.0 %
Precision	89.4 %	95.4 %	95.2 %	95.4 %	90.9 %	100.0 %	**94.4 %**

Table 3-13: Confusion matrix for control scenario of baseline SVM model with raw signals input (SVM trained with reference set, predicting over reference set)

The second setup comparing the own features to the dataset's pre-calculated ones produces the performance figures in table **3-14**. In this complementary scenario precision and recall scores fluctuate moderately, with an overlapping classification of sitting and standing activities showing up. In general, the experimentally preprocessed data and calculated features exhibit certain degree of difference when compared to the ones provided with the dataset.

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**477**	0	37	0	0	0	92.80 %
Walking Upstairs	50	**432**	3	0	1	0	88.89 %
Walking Downstairs	94	3	**353**	0	0	0	78.44 %
Sitting	0	1	0	**326**	197	0	62.61 %
Standing	1	0	0	11	**562**	0	97.91 %
Laying down	0	0	0	0	0	**560**	99.29 %
Precision	76.56 %	99.08 %	89.82 %	96.73 %	73.66 %	100.0 %	**89.31 %**

Table 3-14: Confusion matrix for experimental scenario of baseline SVM model with raw signals input (SVM trained with experimental set, predicting over experimental set)

A final and failed experiment crossing the reference train set with the own calculated test set highlights the extent of the differences between the reference features and the experimental ones, with no feasible prediction because of inter-set values incompatibilities. There seems to be not enough similarities between the structure of train and test sets so as to enable the model to reconstruct the links in the data and generate predictions.

3.5. Summary

Relative success was reached during the attempt to imitate the preprocessing and feature extraction steps of Smartlab's HAR dataset, achieving good accuracy with baseline SVMs resembling the reference model. However this does not mean preprocessing and feature extraction steps performed by the authors of the dataset are repeatable without a complete account of the procedure followed and the assumptions made. Even with the high accuracy (close to 90 %) of the SVM model trained with an approximation of these original steps, the features extracted in the process do differ greatly from the reference sets. A throughout validation of every calculation performed is required, as there seems to be loss of information or structure while transforming the data. Therefore, the Gaussian SVM classifier from section 3.3 will serve as baseline model for the next chapters, for its closer resemblance in form and performance with the reference SVM.

4 Feature learning model for activity recognition

This chapter introduces the architecture and design choices for the proposed classifier model for HAR, which borrows from the field of computer vision the convolutional neural network (convNet or CNN), a deep machine learning technique. This model represents a capable alternative to the SVM classifier adopted by the authors of the Smartlab's HAR dataset (explored in the previous chapter). Moreover, as a representation learning model it greatly reduces the prior knowledge of underlying movement characteristics required to achieve satisfactory performance.

The proposed HAR model is composed of the convolutional-pooling steps and fully connected layers seen in *LeNet-5*-inspired networks [56]. Alternative input representations are considered to further test the model's ability to extract activity-related features: one model type accepts raw accelerometer and gyroscope time-based signals; the other model type is fed the amplitudes computed from the raw data via FFT. The focus here is set on the building blocks of the common CNN architecture, but also on the associated choices realizing the feature learning approach. Next chapter evaluates the proposed CNN, testing it with Smartlab's HAR dataset.

4.1. Model overview

Until very recently most of the research within activity recognition field has depended upon the careful selection and engineering of features. As it was shown in the Smartlab's dataset exploration chapter this process imposes restrictions on the repeatability of research results, particularly when there is no explicit and detailed account of the procedure. In this sense, the feature learning strategy for the proposed HAR model is motivated by the comparatively low scores documented earlier on (while trying to replicate the preprocessing and feature extraction stages of Smartlab's dataset). A representation closer to the inner data structure is expected to be learned by the model itself based on the labeled signals, instead of selecting features according to a prior understanding of the relationship between inertial signals and the set of activities to be recognized.

4.1.1. Primary goals

The design, implementation and deployment of the proposed model seek to obtain a serviceable activity classification out from the raw signals data. This aligns with the global objective of extracting human activity information from sensor data. In a more general sense, the model also serves as a demonstration of the representation learning capabilities of a feature learning approach, confronted here to multi-dimensional input data with varying format.

As for specific goals the supervised model described here attempts to match the performance figures presented in the debut paper of Smartlab's dataset –see table **3-2** in chapter 3–, at the same time improving on the shortcomings of the SVM classifier released alongside. A related goal is to reduce the amount of preprocessing over the raw signals being fed into the pattern recognition model, in contrast to the expert signal modifications incorporated in the reference SVM.

4.1.2. Architecture design choices

This section documents the decisions that ultimately led to the solution adopted for the activity recognition model. It starts off with a brief justification of the model selected, and goes on by presenting a series of considerations for the static and dynamic aspects of the architecture. This way it sets a preamble to the discussion of the actual layer topology, which includes complexity measures (to be used for hyperparameter configuration in the next chapter).

Model selection

Shallow classifiers for HAR demand expert DSP knowledge for their usual reliance on handcrafted features. Other models, like fully connected neural networks, do not scale well when paired to the time-based multi-axial signals proper of motion sensing. And in general, under most traditional neural network models, the high dimensionality of a dataset containing inertial signals would lead to an almost untrainable set of weights.

ConvNets are then better suited to process raw or lightly preprocessed motion signals, because of their explicit assumption of a multi-dimensional image-like input and parameter-sharing scheme . The sparse connections between layers also mean they are able to encode a more efficient function for the forward pass, taking less time to train and compute activity predictions. On the downside, the underlying activation maps are not easily traceable back to the motion signals (as it is the case with visual input classifiers), greatly reducing the understandability of the extracted feature hierarchy.

Preprocessing

The simplified two-step dataset preparation procedure presented in table **4-1** attempts to minimize the preprocessing of inputs. First, raw values are normalized over each movement axis by means of *z-score normalization*, with related formula provided in equation 4-1. This prior transformation is a common practice for neural networks [57]. Afterwards, resulting values are segmented into overlapping windows, to add a basic scale of human activity to the 50Hz raw motion signals. In this windowing step the length and overlap of the window follow closely the guidelines given by the dataset authors, who establish this setup as a reasonable time framing for a human step or elementary action [2].

$$x' = \frac{x - \mu_X}{\sigma_X} \quad \begin{array}{l} \text{where } \mu_X \text{ is the arithmetic mean over the samples and} \\ \sigma_X \text{ is the standard deviation for the same collection.} \end{array} \tag{4-1}$$

Preprocessing step	**Description**
Normalization	Center a_x, a_y, a_z, w_x, w_y and w_z around zero mean with standard deviation equal to one
Windowing	Divide sequential signals into blocks of 128 samples each (2.56 seconds at 50Hz sampling rate), with 50 % overlap

Table 4-1: Simplified preprocessing pipeline for proposed HAR model

Alternative input format

Two types of inputs are considered in order to test the flexibility of the proposed model to process different representations of the same data:

- Raw signals preprocessed through the simplified pipeline introduced above.
- Frequency-domain representation of the already preprocessed raw signals.

These alternative formats correspond to the preprocessing and transformation stages in figure **4-1**, which locates them in relation to the raw data and the actual processing within convolutional model. In particular, the translation of raw input data into the frequency domain proceeds by applying FFT to the normalized multichannel signals within activity windows. A general purpose Hanning window function with 64 points is then rolled over each sequence of 128 real values to zero their edges (making them apt for FFT). After computing the frequency-domain function six sets of amplitudes (one per channel) are finally obtained from each window. This is accomplished by taking the absolute value of the positive frequency bins.

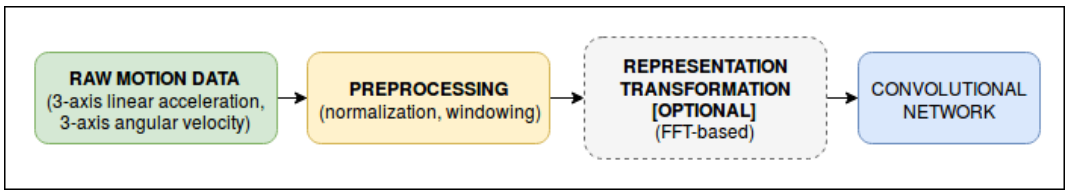

Figure 4-1: Proposed model input and preprocessing stages

Regularization

Two regularization techniques are applied to counter over-fitting at model and experimental setup levels, taking into consideration the relatively small size of the chosen dataset for deep learning purposes (under 12,000 windows after window segmentation, as it will be shown in the next chapter). First, dropout stages are added after fully connected layers in the network. These stages seek to reduce generalization error by dropping units from the neural network during training, with probability $1 - p$ (here p stands for *keep probability* [58]). As an additional measure, early stopping gets incorporated in the setup to preclude training and store the current model once performance metrics meet an experimentally set threshold. The model is then allowed to train for a fixed number of additional epochs, to confirm there is no better model available in later epochs nearby.

Cost and optimization

Softmax cross-entropy between logits and one-hot encoded ground truth labels is used to measure probability error of the mutually exclusive activity classes in table **3-1** of chapter 3: WK, WU, WD, SD, SU, and LD. Softmax itself is an extension of the logistic function to multiclass probability distributions. Equation 4-2 presents probabilities via softmax (S) of class $y = j$ where $j = \{WK, WU, WD, SD, SU, LD\}$ given Z_i, the output of the last layer. Then the softmax probability vector for i^{th} observation can be represented by $S_i = P(y = j|Z_i)$.

$$P(y = j|Z_i) = [S(Z_i)]_j = \frac{e^{Z_{ij}}}{\sum_{p=0}^{k} e^{Z_{ip}}} \tag{4-2}$$

Cross-entropy, (D), computes distance between the output y_i from logits and the actual activity label T_i, as presented in equation 4-3.

$$D(S_i, T_i) = -\sum_{j=1}^{k} T_{ij} log S_{ij} \tag{4-3}$$

The resulting cost function to be minimized via Gradient Descent takes as parameters $\boldsymbol{W}$ (weights) and $\boldsymbol{b}$ (bias) and is shown in equation 4-4.

$$J(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{n} \sum_{i=1}^{n} D(S_i, T_i) \tag{4-4}$$

Adam is selected as Gradient Descent optimization algorithm, given its widespread adoption as of recent and general purpose suitability with minimal hyperparameter tuning. The weight and bias update process by this algorithm is shown in algorithm **4-1**, where $f(\theta) = J(W, b)$. This pseudocode is adapted from Adam's introductory paper [59].

Require: α : step size
Require: $\beta_1, \beta_2 \in [0, 1)$: exponential decay rates for first and second moment estimates
Require: $f(\theta)$: stochastic objective function with parameters θ
Require: θ_0 : initial parameter vector
  m_0,v_0,$t_0 \leftarrow 0$: initialize first and second moment vectors, and time step
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_t f_t(\theta_{t-1})$ (get gradients with respect to f at timestep t)
    $m_t \leftarrow \beta_1.m_{t-1} + (1 - \beta_1).g_t$ (update biased first moment estimate)
    $v_t \leftarrow \beta_2.v_{t-1} + (1 - \beta_2).g_t^2$ (update biased first moment estimate)
    $\hat{m}_t \leftarrow m_t \ / \ (1 - \beta_1^t)$ (compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t \ / \ (1 - \beta_2^t)$ (compute bias-corrected second moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha.\hat{m}_t \ / \ (\sqrt{\hat{v}_t} + \epsilon)$ (compute updated parameters)
  end while
  return θ_t (resulting parameters)

Algorithm 4-1: Adam optimization algorithm (adapted from [59])

Activation

ReLU activation function is deployed for the activation phase of both convolutional and dense layers in the proposed model, following the trend of preferring it over sigmoid function and hyperbolic tangent. This straightforward function presented in equation 4-5 has demonstrated to be a great performer, arguably reflecting the traits and operation of biological neurons [60].

$$f(x) = max(0, x) \tag{4-5}$$

4.2. Layer types

Each one of the basic blocks in the proposed convolutional architecture includes a convolutional layer followed by activation and pooling layers. This topology, depicted in figure **4-2** for a single *convolutional block*, is based on the *LeNet-5* architecture [56] and its details are

presented below. A summary of all parameters and hyperparameters related to each layer type is provided where relevant.

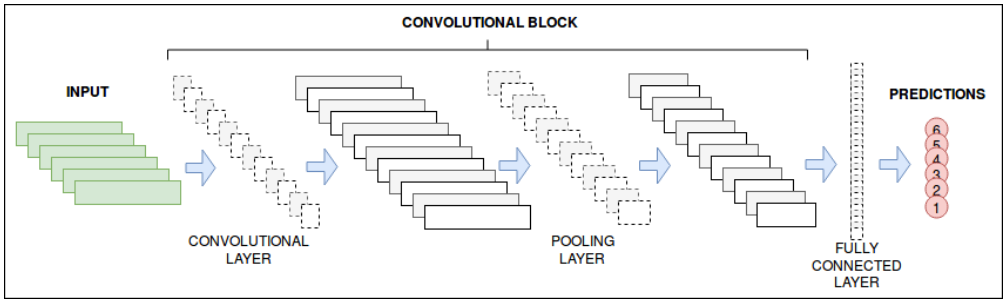

Figure 4-2: Proposed model layer topology (basic instance with one convolutional block)

Input layer (INPUT)

This layer receives the output of preprocessing stage. When the input corresponds to the raw motion representation this layer holds one-dimensional 128 samples by 6-channel-deep windowed chunks of raw data. Here channels $\{a_x, a_y, a_z, w_x, w_y, w_z\}$ match accelerometer and gyroscope triaxial signals of table **3-3** in chapter 3. And for the frequency-domain representation the input dimensions are the same as the output of windowed FFT. In both scenarios of alternative input format a power of two is preferred as input width, because it suits the down sampling steps of the architecture.

Type	Parameter	Description
Hyper-parameters (tunable or preset)	Maximum number of training epochs	Epochs upper limit to cease training when early stopping conditions remain unsatisfied
	Mini-batch size	Size of sample set to be fed at a time to update parameters via Gradient Descent

Table 4-2: Parameters for input layer in proposed HAR model

Convolutional layer (CONV1D)

Every convolutional layer connects locally to the previous layer, either the input layer or the activation layer of another convolutional block. Output volume from convolutional layers in the proposed model is the stack of the 2D activation maps from the convolution of each filter over the input volume, so its depth D_c matches the number of filters. The outputs for feature map j of convolutional layer l with f filters are shown in equation 4-6 [36]. The parameters and hyperparameters related to this layer type within current model are listed in table **4-3**,

with associated formulae in table **4-4**.

$$c_i^{l,j} = \sigma\left(b_j^l + \sum_{f=1}^{F} w_f^{l,j} x_{i+f-1}^{l-1,j}\right) \tag{4-6}$$

Type	**Parameter**	**Description**
Parameters (learnable)	$\boldsymbol{w}^{l,j}$: Weights	Activation thresholds shared across each depth slice (convolutional layer l and feature map j), where each weights set matches filter size
	b_j^l : Bias	Additional trainable terms common to each depth slice, one per slice
Hyper-parameters (tunable or preset)	K_l : Number of filters or output volume depth	Number of convolutional filters or neurons learning from the same input
	F_l : Filter size, kernel size or receptive field	Width of each dimensional convolution to be applied over input
	S_l : Filter stride	Step width to move at a time when sliding filters over the one-dimensional input
	P_l : Zero-padding amount	Zero values added to the margin of input. When set to same padding it allows to preserve the spatial size of output volume between layers

Table 4-3: Parameters for convolutional layers in proposed HAR model

Value	**Formulae**
Input volume	$W_{l-1} x H_{l-1} x D_{l-1}$, with $H_{l-1} = 1$
Output volume $W_l x H_l x D_l$	$W_l = \frac{(W_{l-1} - F_l + 2P_l)}{S_l + 1}$ $H_l = H_{l-1}$ $D_l = K_l$
Number of weights	$(F_l \cdot D_l) \cdot K_l$
Number of bias terms	Same as K_l

Table 4-4: Formulae for dimensions and parameter count in convolutional layers

Each 3D learnable filter in this layer type spans over the full input depth, whether the six inertial signal channels or the corresponding six amplitude channels in the frequency-domain input. Subsequent convolutional layers are expected to detect activity features in an increasing level of abstraction: from sudden changes in linear acceleration and angular velocity up to atomic motion patterns related to activity classes.

Non-linear activation layer

Composed by $ReLU$ units this layer computes activation function $max(0, x)$ over each input x_i coming from the neighboring convolutional layer. Effect of this function is denoted by term σ in equation 4-6. The main purpose of this layer is to extend the capabilities of the convolutional layer to learn complex activity-related features. By definition non-linear activation layers have no parameters nor hyperparameters.

Pooling layer (POOLMAX)

The down-sampling performed by the units in this layer reduces the volume depth by means of a max pooling operation over the full depth of the input, as seen in equation 4-7. The size F_p and stride S_p of the pooling –both hyperparameters– should be small to avoid it being over-destructive over the representation from previous layer. Since there are no parameters apt to be learned from this type of layer, table **4-5** only includes tunable hyperparameters. Formulae of interest to estimate model complexity related to pooling layers are shown in table **4-6**.

$$p_i^{l,j} = max_{r \epsilon R} \left(c_{ixS_p+R}^{l,j} \right) \tag{4-7}$$

Type	Parameter	Description
Hyper-parameters (tunable or preset)	F_p : Pooling size	One-dimensional pooling width to apply max pool function over input
	S_p : Pooling stride	Step width to apply pooling function at a time when sliding it over the one dimensional input

Table 4-5: Parameters for pooling layers in proposed HAR model

Value	Formulae
Input volume	$W_{p-1} x H_{p-1} x D_{p-1}$, with $H_{p-1} = 1$
Output volume $W_p x H_p x D_p$	$W_p = \frac{(W_{p-1} - F_p)}{S_p + 1}$ $H_p = H_{p-1}$ $D_p = D_{p-1}$

Table 4-6: Formulae for dimensions and parameter count in pooling layers

Fully connected layer (DENSE)

In the current model fully connected layers composed of "hidden units" further reduce the dimensionality of the input. But the connections with preceding pooling or fully connected

layer amounts to an extensive set of weights (equal to the number of units times the volume of the previous layer). Also, if the previous layer is not a dense one it has to be "flattened" before interfacing with the dense layers. Table **4-7** summarizes the parameters of fully connected layers, while table **4-8** includes formulae of interest for this layer type.

Type	**Parameter**	**Description**
Parameters (learnable)	$\boldsymbol{w}_i$: Weights	Activation thresholds, one per each connection between unit i and units in previous layer
	b_i :Bias	Additional threshold, unique per unit
Hyper-parameters (tunable or preset)	U_D : Number of units	Neurons within layer
	p : Keep probability	Fraction of units not set to zero at training time

Table 4-7: Parameters for fully connected layers in proposed HAR model

Value	**Formulae**
Input volume	W_{D-1}
Output volume W_D	U_D
Number of weights	$U_D \cdot W_{D-1}$
Number of bias terms	U_D

Table 4-8: Formulae for dimensions and parameter count in dense layers

Logits layer (LOGITS)

This last layer implements a linear function and holds the predictions produced by the model, that is the six activity classes scores. Logits are connected to a prior dense layer via an activation layer, and as a proper fully connected layer share the parameters in table **4-7** (except keep probability) and the formulae in table **4-8**.

4.3. Summary

In this chapter the basic architecture of the proposed model for activity recognition was presented. The basic layout is composed of convolutional blocks within which single convolutional and pooling steps are stacked together. Next chapter evaluates instances of this model using as input the signals in Smartlab's HAR dataset.

5 Experimental evaluation

The previous chapter described the architecture of the proposed model, a convolutional approach with support for alternative input representations: either acceleration and velocity raw signals in the time domain, or the amplitudes of frequency bins extracted from the former. While the basic convolutional blocks are shared among both implementations, they were trained and tuned separately, and their behavior and outcomes differed in some important ways.

The sections below summarize the process and results of testing the feature learning model on the Smartlab's HAR dataset[1]. After introducing a common experimental setup the performance of the two variants of the classifier is detailed in independent sections. In each case the hyperparameter tuning process is followed by the results of model evaluation.

5.1. Common experimental setup

Moving from one input representation (time-domain) to another (frequency-domain) preserved most of the segmentation and basic preprocessing. Therefore, a streamlined common experimental setup mirrors the decision to share a basic architecture between the two implementations, enabling the simplification of the hyperparameter tuning process (as it will be described later). In this section metrics and implementation details common to the varying input implementations are presented.

5.1.1. Metrics

The performance figures included in [2] indicate a value of 96 % overall precision and recall for the SVM reference model of the chosen dataset, with specific scores for *non-dynamic* activities lower than those achieved for *dynamic* activities –see table **3-7** in chapter 3. Both metrics (precision and recall) are retained in this work to evaluate and optimize the recognition model, an they are defined below to establish a common ground:

- *Precision* corresponds to the true positives divided by the sum of true positives and false positives, or equivalently the fraction of relevant detections.

[1] See section 3.1 in chapter 3 for details regarding Smartlab's HAR dataset used in this chapter.

- *Recall* is calculated as the true positives divided by the sum of true positives and false negatives (that is, the fraction of relevant or true items selected).

Precision and recall as presented above are extended to the multi-class scenario of the six activities included in the reference dataset, introduced in table **3-3** of chapter 3. The combination of the two metrics in *mean f1 score* greatly simplifies the performance comparisons for the classifier. Equation 5-1 represents this score calculated over k activity classes.

$$F_{1MACRO-MEAN} = \frac{2}{k} \cdot \sum_{i=1}^{k} \frac{precision_i \cdot recall_i}{precision_i + recall_i} \tag{5-1}$$

In-class scores are also tracked for static and dynamic activities groups, to check for differences in the model's behavior against these broad types of activities. This allows specific evaluation of the separation criteria learned by the model, a topic initially reported as problematic by the dataset owners in relation to their first classifier (and further supported by similar findings in the exploratory analysis of chapter 3). Besides these performance measures, the number of trainable parameters in the tested instances of the HAR model is logged to assess the complexity of the implemented architectures.

5.1.2. Dataset partitioning

For partitioning purposes the 30 subjects within dataset are randomly divided with standard 70/30 ratio into the training set on one side (21 subjects) and the test set on the other (9 subjects). 20 % of the subjects in the training set are then reserved for validation (5 subjects). Afterwards, the complete collection of experimental samples linked to each subject in the sets is recovered from the normalized raw data. This partitioning schema keeps some subjects unknown to the model until it is confronted to them in the test set, enabling learning to focus on classifying activities and not on the specific subjects performing them.

No random shuffle is introduced up to this point to preserve the temporal structure of the motion signals required by preprocessing steps. Table **5-1** lists the resulting sizes for the sets, for one of the test runs (the number of samples per set fluctuates according to the specific subjects included in each partition).

Set	**Subjects**	**Raw samples**	**Activity windows**
Training	16	402,588	6,289
Validation	5	124,192	1,939
Test	9	221,626	3,461
Total	30	748,406	11,689

Table 5-1: Sample partitioning of Smartlab's HAR dataset (single instance of test runs)

5.1.3. Raw signals preprocessing

Over 1.2 million pairs of accelerometer and gyroscope triaxial readings are included in version 2.1 of Smartlab's HAR dataset. Out of these, 748,406 represent the six main activities covered by the original SVM model used for benchmarking purposes in this work, as shown in total cell of table **5-1**. The remaining raw tuples match the postural transitions brought in by the newer model –see section 3.1.1 in chapter 3 for details–, so they are filtered out before partitioning and feeding them into the preprocessing pipeline outlined in chapter 4. Table **5-2** presents the outputs associated to this filtering and to the segmentation and normalization steps.

Preprocessing step	**Output**
Raw values filtering to exclude postural transitions samples	748,406 9-tuples of linear acceleration components $\{\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z\}$ and angular velocity components $\{\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{z}_z\}$, labeled with $\{userID, experimentID, activityID\}$
Window segmentation	11,692 128-sized (2.56 seconds) windows with 50 % overlap
Z-score standardization	No change in dimensions from previous step

Table 5-2: Outcome of preprocessing steps for HAR model

Approximately 13 % of the windows coming out from the segmentation step contained mixed activities, so it was decided to base the activity label for each window on the statistical mode. Comparative tests ran later excluding those heterogeneous windows showed no significant increase nor decrease in resulting performance, thus supporting the previous choice.

5.1.4. Hyperparameter tuning

The same set of hyperparameter values is explored for each type of layer involved, as listed in table **5-3**. This sharing of values follows the reuse of architectural blocks in either variant of the feature learning model. The final result of the tuning process for both implementations is the matter of separate sections within the chapter: 5.2.1 for time-domain input and 5.3.1 for frequency-domain input.

Tunable parameters

A systematic procedure for free-parameter tuning was tried out at first. It could be described as increasingly selective grid-search using well-behaved combinations of the values in table **5-3**. This process began with networks containing a single convolutional block (i.e. convolutional layer plus pooling layer) and one dense layer. For these instances exploration and f1 scores checks took place with all valid combinations of the number of feature maps, the kernel size, and the number of fully connected units. Parameter values for the layout with the

best performing combination of hyperparameters on the hold-out set were selected. Nonetheless, all instances were evaluated on the test set to check for consistency of validation results.

Accumulative test feedback was expected to help build an incrementally good performing model. Unfortunately, no critical hyperparameter nor values were identifiable once this process was extended to deeper network layouts. Hyperparameters seemingly critical for the simpler instances worked differently once additional convolutional and pooling stages were stacked. The behavior of f1 scores revealed no clear path to guide the exploration in the hyperparameter space. So the overall routine was replaced with expensive but practical random search over the ranges in table **5-3**, limiting the number of sampled instances to 100 per model variant (for a total of 200 samples).

All across this exploration preferred combinations of values for hyperparameters were those keeping a balance between validation performance (as measured by f1 scores) and the model's number of parameters. Overtly complex models are deemed too sensible to over-fitting, so they were discarded whenever a simpler drop-in replacement in terms of performance was available.

Layer Type	**Hyperparameter**	**Values**
CONV1D, POOLMAX	Number of convolutional blocks	{1, 2, 3}
CONV1D	Filter size	{[1,2],[1,4], [1,8], [1,16]}
	Depth (filters)	{32, 64, 128}
POOLMAX	Pooling size	{[1,4], [1,8], [1,16]}
DENSE	Number of dense layers	{1,2}
DENSE	Units	{128, 256, 512, 1024}

Table 5-3: Hyperparameter values explored for proposed HAR model

Preset parameters

The hyperparameters listed in table **5-4** did not participate in tuning iterations, so whenever no value is explicitly mentioned for a hyperparameter it should be checked there. Please notice Hanning Window length only applies to model variant with frequency-domain input.

In regard to these preset hyperparameters they are based on either conventional practices within the field (e.g. the use of an spatial extent of 2 for pooling layers, widely regarded as non-destructive) or preliminary test runs. In particular, exponential decay rates for the first and second momentum estimates (β_1 and β_2) of Adam optimization algorithm are set from the start at 0.9 and 0.999, with ϵ (epsilon) and α (learning rate) equal to $1x10^{-8}$ and $1x10^{-3}$ (in all cases as suggested by the algorithm proponents Kingma and Ba [59]).

Hyperparameter	**Type**	**Associated layer type**	**Fixed value**
Hanning Window length	Network structure	INPUT	64
Convolution stride (S_c)	Network structure	CONV1D	1
Zero-padding	Network structure	CONV1D	Same
Pooling width (F_p)	Network structure	POOLMAX	2
Pooling stride (S_p)	Network structure	POOLMAX	2 (same as pooling width to avoid overlapping pooling)
Keep rate	Training setup	DENSE	0,2
Layer weights initial value	Training setup	CONV1D, DENSE	Randomly sampled from normal distribution with $\delta = 1x10^{-1}$
Loss function	Training setup	Model-wide	Softmax cross-entropy with logits layer
Optimization function	Training setup	Model-wide	Adam
Learning rate (α)	Training setup	Model-wide	$1x10^{-3}$
First moment estimate (β_1)	Training setup	Model-wide	0,9
Second moment estimate (β_1)	Training setup	Model-wide	0,999
Epsilon (ϵ)	Training setup	Model-wide	$1x10^{-8}$
Maximum number of training epochs	Training setup	Model-wide	500
Mini-batch size	Training setup	Model-wide	64

Table 5-4: Preset values for hyperparameters excluded from tuning process of proposed HAR model

5.1.5. Additional considerations

Initial runs achieved rapid convergence, so early stopping [58] was integrated in experimental setup to save time and computational resources. Stopping criteria comprises a 90 % minimum threshold for overall validation accuracy, at least 100 training epochs, and 50 trial epochs without improvement in validation loss (i.e. *patience*). For each explored instance the best performing model up to early stopping or by reaching the maximum of epochs was selected for classification over the test set.

In what follows process and results of the two implementations of the proposed model are covered in independent sections, with a closing review of general findings.

5.2. Results for implementation with raw values input

This variant of the HAR model responds to the global objective of a simplified feature learning approach to activity recognition. As such it tests the capability of a feature learning model to perform well with raw input, without requiring handcrafted purpose-built features. To this end, tuning and testing cycles pursue performance scores on par with the benchmark SVM model, with the less complex architecture feasible.

5.2.1. Hyperparameter sampling results

Taken as a whole, the set of instances sampled for this model implementation required few training epochs to achieve competitive performance scores (43 out of the 100 instances satisfied early stopping criteria within 150 epochs, and overall just 4 instances reached the maximum of epochs). Mean f1 score was very high on average at 0.935. However, in-class distinction between static activities showed comparatively lower performance in every instance of this model, with specific layouts departing from this trend only by a small margin.

At first, given the presence of various sources of randomness (i.e. random weight initialization, dropout, Stochastic Gradient Descent) the tuning process did not reveal any clear trend towards optimal values for any given set of hyperparameters. In particular, under those circumstances there was no correlation between the chosen complexity measure (number of trainable parameters) and the performance of the networks.

After careful examination some general layouts were identified as better adapted to capture the underlying activity representation in motion signals (as measured by mean f1 score). In this respect top performing instances favored a number of feature maps over 128, in disregard of the number of stacked convolutional blocks. The rare cases where a network with just 32 feature maps performed well corresponded to layouts with a high number of fully connected units (1,024). This higher count of weights in an otherwise shallow model is problematic for the increased risk of over-fitting.

During exploration the two worst performers were layouts unable to classify any sample into walking downstairs class and almost any sample into walking upstairs, wrongly labeling these activities as walking. These instances, whose hyperparameter values are briefly listed in table **5-5**, achieved overall accuracies under 70 % because of ill-defined precision for the walking downstairs activity. Table **5-6** presents the related scores (here *NA* denotes undefined scores).

Noticeably, even the lowest performing classifiers were able to correctly label samples from the laying down class with almost complete absence of type I and type II errors.

Layer	**Low performing (LP) convNet #1**	**Low performing (LP) convNet #2**
CONV1D (C1)	128 x [1,16] filters	64 x [1,16] filters
CONV1D (C2)	64 x [1,2] filters	128 x [1,2] filters
CONV1D (C3)	64 x [1,2] filters	Layer not in instance
DENSE (D1)	128 units	256 units
DENSE (D2)	128 units	128 units

Table 5-5: Layout details for low performing (LP) HAR models with raw values input

Model (raw)	**Acc. (%)**	**mean f1**	**f1-dyn**	**f1-ndyn**	**f1-WK**	**f1-WU**	**f1-WD**	**f1-SD**	**f1-SU**	**f1-LD**
LP conv-Net #1	66.15	NA	NA	0.936	0.502	0.004	NA	0.903	0.907	0.998
LP conv-Net #2	64.33	NA	NA	0.902	0.504	0.007	NA	0.847	0.860	0.998

Table 5-6: Scores for low performing (LP) HAR models with raw values input

The performance of networks with raw values input is reflected in figure **5-1**, which plots the behavior of hyperparameters found to be related to overall f1 scores.

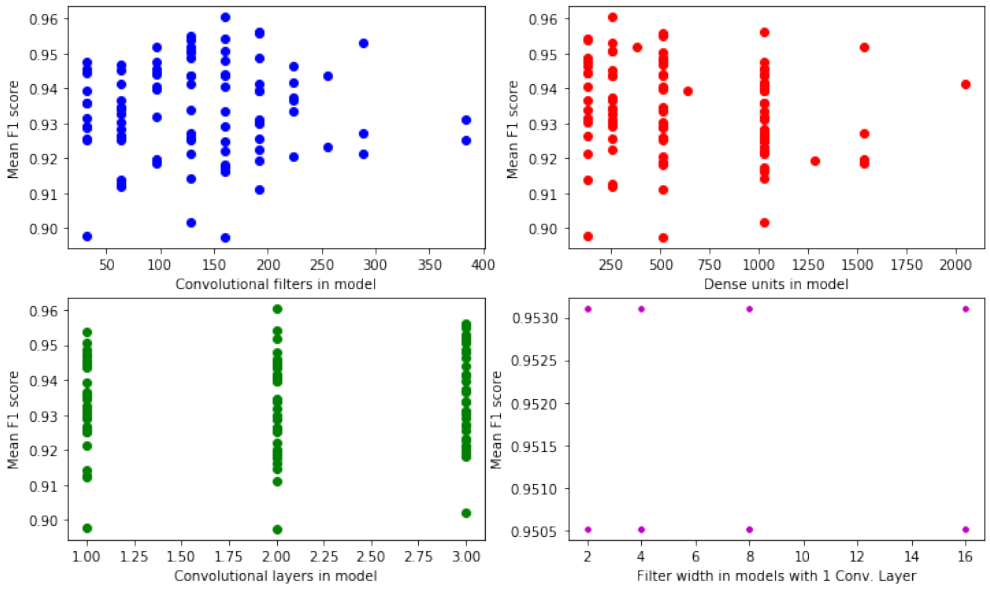

Figure 5-1: Parameters related to overall performance of HAR model with raw values input

5.2.2. Classification results

With a mean f1 score of 0.961 and 95.93 % overall accuracy the best performing network with raw input contains two convolutional blocks with 128 and 32 feature maps respectively, and a sole densely connected layer with 256 units. Kernel width for this instance varies from 2 to 8 between blocks. Complete setup for this instance is registered in table **5-7**, with the complete confusion matrix reproduced in table **5-8**. The associated global and per-class performance in table **5-9** are very close to scores from the reference and baseline SVM models, and reveal the same traits and limitations for the dynamic and non-dynamic sets of activities. In particular, f1 score of non-dynamic activities gets pulled down by the classification overlap between standing and sitting activities, whereas dynamic activities are classified with a consistently high level of precision and recall (reflected in an f1 score of 0.979).

Layer	**Hyperparameter**	**Value**	**Trainable parameters**
CONV1D (C1)	Number of filters	123	1,664
	Filter size	[1,2]	
CONV1D (C2)	Number of filters	32	32,800
	Filter size	[1,8]	
DENSE (D1)	Number of units	256	262,400
	Keep probability	0.2	
LOGITS	Number of units	6	1,542
	Trainable parameters in model		298,406

Table 5-7: Hyperparameter values for best performing HAR model with raw values input

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**550**	0	1	0	0	0	99.82 %
Walking upstairs	4	**547**	5	0	0	0	98.38 %
Walking downstairs	7	9	**505**	0	0	0	96.93 %
Sitting	2	4	0	**579**	23	8	93.99 %
Standing	2	9	0	71	**589**	0	87.78 %
Laying down	0	0	0	0	0	**651**	100.00 %
Precision	97.35 %	96.13 %	98.83 %	89.08 %	96.24 %	98.79 %	96.1 %

Table 5-8: Confusion matrix for best performing HAR model with raw values input

Model	**Acc. (%)**	**mean f1**	**f1-dyn**	**f1-ndyn**	**f1-WK**	**f1-WU**	**f1-WD**	**f1-SD**	**f1-SU**	**f1-LD**
Best raw conv-Net	95.93	0.961	0.979	0.942	0.986	0.972	0.979	0.915	0.918	0.994
Reference SVM	96.37	0.964	0.975	0.953	0.974	0.969	0.982	0.922	0.936	1.000
Baseline Gaussian SVM	94.70	0.945	0.935	0.954	0.933	0.920	0.954	0.926	0.937	1.000

Table 5-9: Comparative scores for best performing HAR model with raw values input

Remarkably, one of the simpler layouts tested (in terms of depth and number of convolutional filters and dense units) reached a mean f1 score of 0.931. This model had 32 feature maps in a single convolutional block, a filter width of 4, and 128 units in its single dense layer. Its lower specific scores matched the global trend of misclassified non-dynamic samples, labeling 30 % of sitting samples as standing. Table **5-10** contains details on the layout of this simple HAR model with 263,846 trainable parameters; tables **5-11** and **5-12** summarize its scores.

Layer	**Hyperparameter**	**Value**	**Trainable parameters**
CONV1D (C1)	Number of filters	32	800
	Filter size	[1,4]	
DENSE (D1)	Number of units	128	262,272
	Keep probability	0.2	
LOGITS	Number of units	6	774
	Trainable parameters in model		263,856

Table 5-10: Hyperparameter values for better performing simple HAR model with raw values input

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**533**	5	12	0	0	1	96.73 %
Walking upstairs	8	**543**	5	0	0	0	97.66 %
Walking downstairs	21	9	**491**	0	0	0	94.24 %
Sitting	3	3	0	**457**	152	1	74.19 %
Standing	4	9	2	11	**644**	1	95.98 %
Laying down	0	0	0	0	0	**651**	100.00 %
Precision	96.37 %	95.43 %	96.27 %	97.65 %	80.90 %	99.54 %	93.91 %

Table 5-11: Confusion matrix for better performing simple HAR model with raw values input

Model	Acc. (%)	mean f1	f1-dyn	f1-ndyn	f1-WK	f1-WU	f1-WD	f1-SD	f1-SU	f1-LD
Simple raw convNet	93.07	0.931	0.957	0.906	0.952	0.965	0.952	0.843	0.878	0.998
Reference SVM	96.37	0.964	0.975	0.953	0.974	0.969	0.982	0.922	0.936	1.000
Baseline Gaussian SVM	94.70	0.945	0.935	0.954	0.933	0.920	0.954	0.926	0.937	1.000

Table 5-12: Comparative scores for better performing simple HAR model with raw values input

5.3. Results for implementation with frequency-domain input

The alternative implementation of proposed model HAR model modifies the format of the input to amplitude. It achieves this by means of an FFT-based transformation interposed between the preprocessed and segmented raw signals at one end, and the network on the other. What is tested here is the adaptability of the CNN to varying input formats (i.e. the independence of the learned representation of activities from the low level data format).

5.3.1. Hyperparameter sampling results

The general experience of this second round of random sampling differed in a few key points from the previous iteration (for the time-based input model). Early convergence was common as before, but 16 % of the models sampled reached exhaustion of training epochs. Good performance was easy to reach, but with a mean f1 score of 0.877 on average scores were lower than those for instances with time-based raw input. Also, there were more cases of extremely under-performing classification of walking activities (*WK*, *WU* and *WD*).

Some specific trends emerged in the form of the layouts preferred by best performing instances with amplitude input. Most of these top performing networks had single convolutional and pooling layers paired to one or two dense layers. In fact, the best performing instance –as it will be covered in next section– happened to be one very closely related to the simple layout detailed in table **5-10**. Frequency-domain input networks with the highest scores also favored narrow receptive fields sized between $[1, 2]$ and $[1, 8]$ for the first convolutional step.

The two worst f1 scores came from networks unable to label any sample as walking downstairs during testing, as can be seen in tables **5-13** and **5-14**. One of these instances clustered

walking upstairs and walking downstairs samples together into walking class, demonstrating the same classification overlap documented earlier. Still, the other bottom performer misclassified walking downstairs and sitting samples as pertaining to standing class, effectively overriding the separation criteria between dynamic and non-dynamic activities. This points to a situation of the model underfitting the data points, as it can be seen in plots of accuracy and loss as a function of training epochs in figure **5-2**.

Layer	**Low performing (LP) convNet #3**	**Low performing (LP) convNet #4**
CONV1D (C1)	128 x [1,2] filters	128 x [1,16] filters
CONV1D (C2)	128 x [1,8] filters	Layer not in instance
DENSE (D1)	256 units	512 units
DENSE (D2)	256 units	128 units
Trainable parameters in model	724,742	2,176,518

Table 5-13: Layout details for low performing (LP) HAR models with amplitude input

Model (ampl. input)	**Acc. (%)**	**mean f1**	**f1-dyn**	**f1-sta**	**f1-WK**	**f1-WU**	**f1-WD**	**f1-SD**	**f1-SU**	**f1-LD**
LP convNet #3	61.10	NA	NA	0.834	0.506	0.042	NA	0.733	0.776	0.993
LP convNet #4	59.25	NA	NA	0.490	0.799	0.807	NA	0.003	0.501	0.966

Table 5-14: Scores for low performing (LP) HAR models with amplitude input

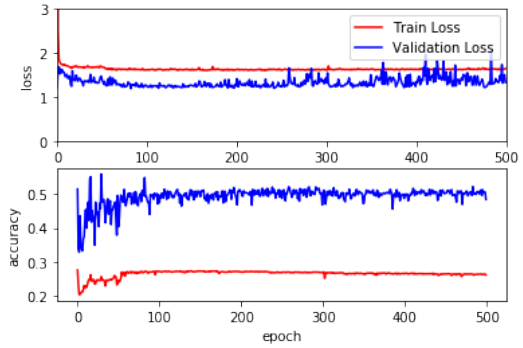

Figure 5-2: Training and validation accuracy and loss plots for lowest performing HAR model with amplitude input

Figure **5-3** plots hyperparameters found to be critical for overall performance of HAR models with raw input. As previously stated smaller convolutional filters perform better in networks with one convolutional block. Also, scores benefit from a total number of filters under 200 and a number of dense units less or equal to 1,000.

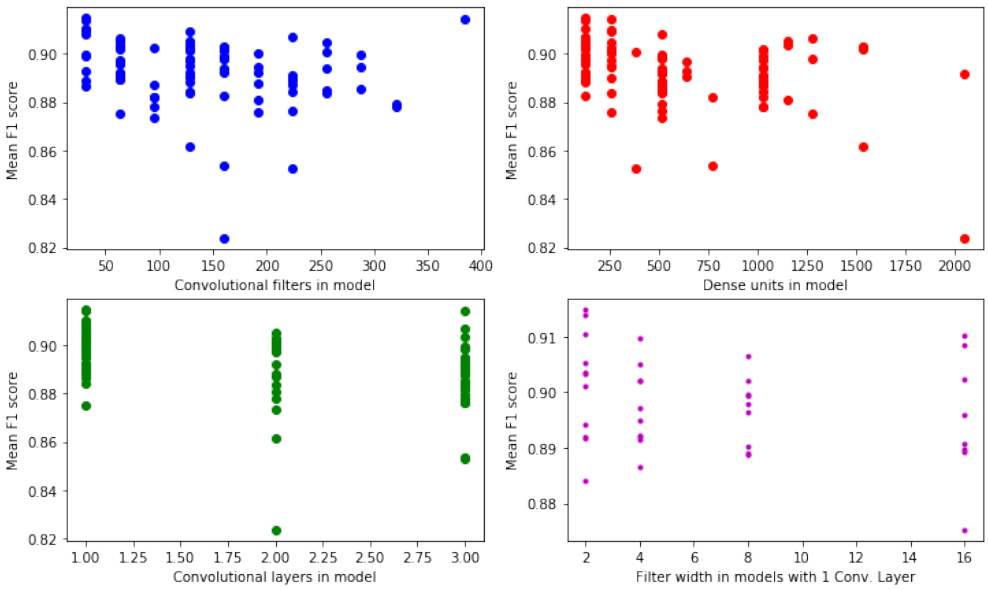

Figure 5-3: Parameters related to overall performance of HAR models with amplitude input

5.3.2. Classification results

The best performing instance with frequency-domain input contains a single convolutional block and single dense layer, as shown in table **5-15**. This model has the lowest number of filters explored (32), the most conservative kernel size (2-units wide), and the smallest dense layer considered (128 units). The associated confusion matrix in table **5-16** and comparative results in table **5-17** match the general trend of misclassifying walking upstairs and walking downstairs samples.

Layer	**Hyperparameter**	**Value**	**Trainable parameters**
CONV1D (C1)	Number of filters	32	416
	Filter size	[1,2]	
DENSE (D1)	Number of units	128	131,200
	Keep probability	0.2	
LOGITS	Number of units	6	774
	Trainable parameters in model		132,390

Table 5-15: Hyperparameter values for best performing HAR model with amplitude input

	WK	WU	WD	SD	SU	LD	**Recall**
Walking	**540**	1	7	3	0	0	98.00 %
Walking upstairs	3	**513**	35	0	5	0	92.27 %
Walking downstairs	22	17	**482**	0	0	0	92.51 %
Sitting	1	1	0	**509**	103	2	82.63 %
Standing	3	1	0	102	**562**	3	83.76 %
Laying down	0	0	0	1	0	**650**	99.85 %
Precision	99.12 %	96.25 %	97.14 %	88.38 %	94.30 %	99.23 %	95.74 %

Table 5-16: Confusion matrix for best performing HAR model with amplitude input

Model	**Acc. (%)**	**mean f1**	**f1-dyn**	**f1-ndyn**	**f1-WK**	**f1-WU**	**f1-WD**	**f1-SD**	**f1-SU**	**f1-LD**
Best amplitude convNet	93.31	0.915	0.943	0.887	0.964	0.942	0.922	0.827	0.838	0.995
Reference SVM	96.37	0.964	0.975	0.953	0.974	0.969	0.982	0.922	0.936	1.000
Baseline Gaussian SVM	94.70	0.945	0.935	0.954	0.933	0.920	0.954	0.926	0.937	1.000

Table 5-17: Comparative scores for best performing HAR model with amplitude input

5.4. Discussion

The end-to-end effort documented in this text illustrates the properties making feature learning –and convNets in particular– suitable for research in human activity recognition. To do so the current work first compares the representation learning approach to conventional supervised learning with handcrafted features. Then it proceeds to implement and evaluate a CNN-based HAR model. Across this whole process the same dataset (Smartlab's HAR dataset) is used to establish a common ground for comparisons.

On this matter chapter 3 highlighted the extent to which feature engineering is a complex and expert process: in absence of a thorough account of the original procedure and all related choices it is difficult to reproduce results in research reports. This was the case with the preprocessing and feature extraction stages outlined in [26] whose outcomes with respect to features and performance levels were partially imitated here. 134 out of 561 features were extracted from the inertial raw data in HAR dataset, and prediction performance of an SVM model similar to the reference classifier achieved a global score within 7 % of the accuracy claimed in the original paper. However, the gap for per-activity scores was much higher (up

to 20 % for *walking* activity), and the generated features were not comparable to the ones calculated by the authors (provided also as part of the reference dataset). Moreover, trying to mix the reference features with own features in a training/testing cycle of the SVM ended up generating data incompatibility errors. So the results were not really reproduced.

Following this a CNN model for raw HAR signals was introduced, inspired by the now archetypal LeNet-5. As part of it a lightweight preprocessing pipeline infused the raw signals with a temporal scale characteristic of human activities. At training time systematical hyperparameter tuning remained problematic, and the random search used as a substitute revealed the effect of randomness sources integrated within the model: weight initialization, dropout and Stochastic Gradient Descent made it difficult to isolate critical hyperparameters and good-behaving values for them. Nonetheless, general guidelines were identified and performance figures of the top performing instances were comparable to those of the reference and baseline models, with no feature engineering nor DSP expertise involved.

Overall, the experimental evaluation of the proposed HAR model with Smartlab's dataset produced high scores comparable to those from the reference SVM model, with a slight decrease in scores for the model variant with frequency-domain input (average f1 score went down from 0.935 to 0.877). Surprisingly, one of the simplest raw-input instances evaluated –with single convolutional and pooling steps– achieved a mean f1 score in excess of 0.92 over Smartlab's HAR dataset, serving as a proof of the suitability of CNN models for HAR.

The limitations in the representation learned by instances of the proposed HAR model came in the form of classification overlaps between walking downstairs and walking upstairs activity classes. This misclassification case was previously documented for the reference SVM in [26] and reappeared in the baseline models of chapter 3. In this regard, a similar exploration of the same dataset identified as source of these comparatively low scores the divergent motion patterns of a single subject [36].

6 Conclusions and future work

6.1. Conclusions

Practical outcome from this work with value of reuse for research in activity recognition is summarized in the following statements:

- Inertial signals from wearable accelerometers and gyroscopes –like those in Smartlab's dataset– represent a rich source of raw activity data, able to capture from a first-person perspective the finesse of motion changes in elementary human actions. Nonetheless, a light preprocessing stage with a sliding window is still recommended to incorporate a scale of human activity in the raw data.

- Handcrafted feature extraction for HAR is a hard-to-replicate procedure, for the many expert choices lacking proper documentation within reports of results. This was exemplified by the Smartlab's HAR dataset, where the unusual availability of both raw data and features allowed a direct comparison of results between the original procedure and the attempted replica.

- In contrast to conventional feature engineering, representation discovery greatly simplifies the preprocessing and feature extraction stages of HAR with no impact on performance, leveling the field for researchers not trained in DSP. In particular, by replacing a wide array of highly specific time-domain and frequency-domain measures with a hierarchy of *activity-domain* features discovered by the convNet, performance comparable to an SVM model with handcrafted features was easily achieved.

- Deep learning is justified as the current model family of choice for HAR. As demonstrated by the CNN deployed here, this approach answers the question of how to extract meaningful activity information from sensor data: don't impose an external ad-hoc representation. Deep learning effectively substitutes shallow activity patterns with more meaningful data-based features, even when a global function mapping the time-domain signals to the frequency domain is interposed.

- Random exploration of hyperparameters and controlling sources of randomness where feasible (e.g. seeds for weight initialization) emerges as the best strategy for hyperparameter tuning of CNN models in HAR. This limited suitability to systematic approaches

remains a minor inconvenience when the convincing performance right from the start is considered.

- On the downside of feature learning approaches for HAR, the representation extracted out of the set of activities is not readily traceable back to traits in the raw inertial signals (contrasting with networks with image-based inputs, where activation maps can be explicitly linked to RGB inputs). This is at least partially related to the non-trivial interpretation of forward pass activation maps and weights, something required to understand the inner representation of activities being learned.

6.2. Future work

Additional inertial data could help train a more robust model able to compensate for the sensitivity of models to unusual motion traits of single subjects. In this sense there is a growing catalog of public datasets containing inertial signals, some of them re-purposing smart phones as sensing devices just like Smartlab's HAR dataset. The development of a mobile application to capture additional data from the embedded accelerometer and gyroscope (if available) could further expand the training samples. Alternatively, the integration of unsupervised pre-training to the proposed HAR model could lessen the dependence on labeled data and enable online recognition.

Another pending task is related to the underlying MEMS technology used for accelerometer sensors in mobile devices. They are deemed as systematically inaccurate on a per-device basis, to the point of generating a unique signature apt to be used for subject identification purposes [61]. While this is domain-specific and contradicts somehow a simplified preprocessing pipeline, further research on this low-level bias not related to motion patterns could contribute to improve the performance of current HAR models.

Bibliography

[1] L. Deng, D. Yu, *et al.*, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones.," in *ESANN*, 2013.

[3] M. Weiser, "The computer for the 21st century.," *Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3–11, 1999.

[4] P. Dourish, "What we talk about when we talk about context," *Personal and ubiquitous computing*, vol. 8, no. 1, pp. 19–30, 2004.

[5] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, "Sensor-based activity recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 790–808, 2012.

[6] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[7] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[8] M. P. Lawton and E. M. Brody, "Assessment of older people: self-maintaining and instrumental activities of daily living," *The gerontologist*, vol. 9, no. 3_Part_1, pp. 179–186, 1969.

[9] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall, "A survey on human motion analysis from depth data," in *Time-of-flight and depth imaging. sensors, algorithms, and applications*, pp. 149–187, Springer, 2013.

[10] L. E. Dunne, P. Walsh, B. Smyth, and B. Caulfield, "Design and evaluation of a wearable optical sensor for monitoring seated spinal posture," in *Wearable Computers, 2006 10th IEEE International Symposium on*, pp. 65–68, IEEE, 2006.

[11] P. Lukowicz, F. Hanser, C. Szubski, and W. Schobersberger, "Detecting and interpreting muscle activity with wearable force sensors," in *International Conference on Pervasive Computing*, pp. 101–116, Springer, 2006.

[12] N. Oliver and F. Flores-Mangas, "Healthgear: a real-time wearable system for monitoring and analyzing physiological signals," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp, IEEE, 2006.

[13] T. Westeyn, P. Presti, and T. Starner, "Actiongsr: A combination galvanic skin response-accelerometer for physiological measurements in active environments," in *Wearable Computers, 2006 10th IEEE International Symposium on*, pp. 129–130, IEEE, 2006.

[14] T. Linz, C. Kallmayer, R. Aschenbrenner, and H. Reichl, "Fully untegrated ekg shirt based on embroidered electrical interconnections with conductive yarn and miniaturized flexible electronics," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp, IEEE, 2006.

[15] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, *et al.*, "The mobile sensing platform: An embedded activity recognition system," *IEEE Pervasive Computing*, vol. 7, no. 2, 2008.

[16] L. Atallah and G.-Z. Yang, "The use of pervasive sensing for behaviour profiling —a survey," *Pervasive and Mobile Computing*, vol. 5, no. 5, pp. 447–464, 2009.

[17] W. He, Y. Guo, C. Gao, and X. Li, "Recognition of human activities with wearable sensors," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 108, 2012.

[18] N. Eagle and A. S. Pentland, "Reality mining: sensing complex social systems," *Personal and ubiquitous computing*, vol. 10, no. 4, pp. 255–268, 2006.

[19] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods of information in medicine*, vol. 48, no. 5, p. 480, 2009.

[20] O. Brdiczka, M. Langet, J. Maisonnasse, and J. L. Crowley, "Detecting human behavior models from multimodal observation in a smart home," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 4, pp. 588–597, 2009.

[21] D. Guan, T. Ma, W. Yuan, Y.-K. Lee, and A. Jehad Sarkar, "Review of sensor-based activity recognition systems," *IETE Technical Review*, vol. 28, no. 5, pp. 418–433, 2011.

[22] J. A. Ward, P. Lukowicz, G. Troster, and T. E. Starner, "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1553–1567, 2006.

[23] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *International conference on pervasive computing*, pp. 158–175, Springer, 2004.

[24] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg, "A scalable approach to activity recognition based on object use," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, IEEE, 2007.

[25] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *arXiv preprint arXiv:1707.03502*, 2017.

[26] D. Anguita, A. Ghio, L. Oneto, F. X. Llanas Parra, and J. L. Reyes Ortiz, "Energy efficient smartphone-based activity recognition using fixed-point arithmetic," *Journal of universal computer science*, vol. 19, no. 9, pp. 1295–1314, 2013.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[29] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[30] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 1729, 2011.

[31] N. Y. Hammerla, J. Fisher, P. Andras, L. Rochester, R. Walker, and T. Plötz, "Pd disease state assessment in naturalistic environments using deep learning.," in *AAAI*, pp. 1742–1748, 2015.

[32] N. D. Lane, P. Georgiev, and L. Qendro, "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 283–294, ACM, 2015.

[33] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[34] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, "Learning human identity from motion patterns," *IEEE Access*, vol. 4, pp. 1810–1820, 2016.

[35] A. Karpathy, "Stanford university cs231n: convolutional neural networks for visual recognition," *URL: http://cs231n.stanford.edu/syllabus.html*, 2017.

[36] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

[37] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition.," in *IJCAI*, pp. 3995–4001, 2015.

[38] S. S. Intille, K. Larson, J. Beaudin, J. Nawyn, E. M. Tapia, and P. Kaushik, "A living laboratory for the design and evaluation of ubiquitous computing technologies," in *CHI'05 extended abstracts on Human factors in computing systems*, pp. 1941–1944, ACM, 2005.

[39] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Wearable Computers (ISWC), 2012 16th International Symposium on*, pp. 108–109, IEEE, 2012.

[40] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy, "Distributed recognition of human actions using wearable motion sensor networks," *Journal of Ambient Intelligence and Smart Environments*, vol. 1, no. 2, pp. 103–115, 2009.

[41] F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran, "Guide to the carnegie mellon university multimodal activity (cmu-mmac) database," *Robotics Institute*, p. 135, 2008.

[42] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pp. 233–240, IEEE, 2010.

[43] M. Zhang and A. A. Sawchuk, "Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1036–1043, ACM, 2012.

[44] "Uci machine learning repository." `https://archive.ics.uci.edu/ml`. Accessed: 2018-01-30.

[45] J.-L. Reyes-Ortiz, L. Oneto, A. Ghio, A. Samá, D. Anguita, and X. Parra, "Human activity recognition on smartphones with awareness of basic activities and postural transitions," in *International Conference on Artificial Neural Networks*, pp. 177–184, Springer, 2014.

[46] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[47] "Uci machine learning repository." `https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones`. Accessed: 2018-02-16.

[48] "Uci machine learning repository." `http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions`. Accessed: 2018-02-16.

[49] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[50] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Training computationally efficient smartphone–based human activity recognition models," in *International Conference on Artificial Neural Networks*, pp. 426–433, Springer, 2013.

[51] N. Wang, E. Ambikairajah, N. H. Lovell, and B. G. Celler, "Accelerometry based classification of walking patterns using time-frequency analysis," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 4899–4902, IEEE, 2007.

[52] A. Sama, D. E. Pardo-Ayala, J. Cabestany, and A. Rodríguez-Molinero, "Time series analysis of inertial-body signals for the extraction of dynamic properties from human gait," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1–5, IEEE, 2010.

[53] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International Conference on Pervasive Computing*, pp. 1–17, Springer, 2004.

[54] A. Kos, S. Tomažič, and A. Umek, "Suitability of smartphone inertial sensors for real-time biofeedback applications," *Sensors*, vol. 16, no. 3, p. 301, 2016.

[55] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[57] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[60] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[61] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pp. 306–311, IEEE, 2010.