Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

---

Tesi di Laurea Magistrale in Ingegneria Informatica

Multimodal Sensor Data Fusion for Activity Recognition using Deep Neural Network Ensembles

Candidato
Francesco Pegoraro

Relatori
Prof. Andrew D. Bagdanov
Prof. Enrico Vicario

Correlatore
Prof. Chris Nugent

Anno Accademico 2017/2018

Dedication

I dedicate this thesis to my family and many friends.

A special feeling of gratitude to my loving parents, Paola, and Roberto whose words of encouragement and push for tenacity ring in my ears.

I also dedicate this dissertation to the friends who have supported me throughout my studies. I will always appreciate all they have done, especially my colleagues, for contributing develop my skills, for the many hours of fun, and for helping me to become who I am.
I dedicate this work and give special thanks to my best friend and lover Mariavittoria for being there for me throughout the entire time.

Contents

Abstract

Activity recognition systems deployed in smart homes are characterized by their ability to detect Activities of Daily Living (ADL) in order to improve assistance. Such solutions have been adopted by smart homes in practice and have delivered promising results for improving the quality of care services for elderly people and responsive assistance in emergency situations.

We propose a novel framework for activity Learning able to extract features from multiple types of sensors located in a sensor-rich environment and make a prediction based on their output. Both CNN and LSTM architectures will be used in a multi-input manner to enhance the classification accuracy making use of all the data sources. In addition, an average ensemble of the same model, but with different window sizes is used to overcome the choosing size problem and to learn to discern activities of different complexity. We will explain the phases of data processing and discuss the framework in detail, exploring the unprecedented use of multi-input Neural Networks in AR and the *temporal ensemble,* motivating the choices of design and evaluating its ability to generalize in different scenarios.

We evaluated our framework on the UCAmI Cup dataset, which consists in the recognition of 24 activities of daily living and contains data collected from four data sources: binary sensors, an intelligent floor, proximity and acceleration sensors. Our results show that our framework outperforms com-

peting machine and deep learning techniques by 13%. We demonstrate that the framework can be applied to homogeneous sensor modalities, but can also fuse multimodal sensors to improve performance. We characterise key architectural hyperparameters' influence on performance to provide insights about their optimisation.

Abstract (italiano)

I sistemi per Activity Recognition sviluppati nelle smart home, sono caratterizzati dalla loro capacità di rilevare le attività quotidiane (Activities of Daily living, ADL) al fine di migliorare l'assistenza. Tali soluzioni sono state adottate nelle smart home nella pratica e hanno prodotto risultati promettenti per migliorare la qualità dei servizi di assistenza per gli anziani e in situazioni di emergenza.

In questa tesi, proponiamo un nuovo framework per l'Activity Learning, in grado di estrarre features da più tipi di sensori situati in un ambiente sensor-rich e fare una previsione basata sul loro output. Entrambe le architetture, CNN e LSTM, saranno utilizzate in modo multi-input per migliorare l'accuratezza della classificazione, facendo uso di tutte le fonti di dati presenti nell'ambiente. Inoltre, un average ensemble dello stesso modello, ma con dimensioni di finestre diverse, viene utilizzato per superare il problema della scelta della dimensione noto nei problemi Time Series e per imparare a discernere attività di diversa complessità. Spiegheremo le fasi dell'elaborazione dei dati e discuteremo dettagliatamente il framework, esplorando l'uso senza precedenti delle reti neurali multi-input in AR e del temporal ensemble, motivando le scelte di progettazione e valutando la sua capacità di generalizzare in diversi scenari.

Al fine di valutare le performance del nostro framework, l'abbiamo testato

sulla competizione UCAmI Cup, che consiste nel riconoscimento di 24 classi di attività della vita quotidiana e contiene dati raccolti da quattro fonti di dati: sensori binari, sensori di prossimità, di pressione e di accelerazione. I nostri risultati mostrano che il nostro framework, supera le tecniche di apprendimento degli altri partecipanti del 13%. Dimostriamo che la struttura può essere applicata a modalità di sensore omogenee, ma può anche fondere sensori multimodali per migliorare le prestazioni. Individuiamo l'influenza degli iperparametri nella nostra architettura sulle prestazioni per fornire approfondimenti sulla loro ottimizzazione.

Chapter 1

Introduction

The total population in the EU is projected to increase from 511 million in 2016 to 520 million in 2070. However, the working-age population (people aged between 15 and 64) will decrease significantly from 333 million in 2016 to 292 million in 2070. The old-age dependency ratio (people aged 65 and above relative to those aged 15 to 64) in the EU is projected to increase by 21.6 percentage points, from 29.6% in 2016 to 51.2% in 2070. This implies that the EU would go from having 3.3 working-age people for every person aged over 65 years to only two working-age persons [1].

The aging of Europe's population will also have significant implications for its cost. Overall in the EU, the total cost of aging (public spending on pensions, health care, long-term care, education, and unemployment benefits), is expected to increase by 1.7 percentage points to 26.7% of GDP between 2016 and 2070.

Long-term care and health care costs are expected to contribute the most to the rise in age-related spending, increasing by 2.1 percentage points. Public spending on pensions is expected to increase until 2040 before returning close to current levels by 2070. Education expenditure is projected to remain

unchanged until 2070.

What all this evidence points to is an unavoidable, worldwide, increase in the age profile for humankind and therefore an increase in the prevalence of age-related diseases, including dementia.

As pictured by those data, the growth of the elderly population will become unsustainable. Moreover, when referring to elderly care costs, it is worth noticing that these problems are not only applicable to elderly patients (and thus to cure and healthcare costs), but also to non-patient, healthy elderly who risk losing their autonomy and independence.

One of the most common co-morbidities of elderly persons is dementia, which is diagnosed by a set of progressive symptoms such as aphasia (loss of language function), apraxia (loss of the ability to perform intentional movements), as well as agnosia (loss of the ability to recognise objects), and problems with abstract thinking and complex behaviour. The most common form of dementia is generally accepted to be Alzheimer's disease (AD). In the early stages, a person with dementia (PwD) needs memory support, help with regular daily activities and social contact. In the mild stage of the disease, special medication and medical care become necessary. Care and management continue and are progressively more intensive as the disease progresses until it reaches the most severe stages. One of the symptoms of AD is a tendency to wander from the home and at some point sleep eventually enters a phase shift with wakeful nights leading to night-time wandering, which is usually the precursor to institutionalization. Technological solutions aimed at compensating for disabilities, such as memory problems and daily activities demonstrate that people with mild to moderate dementia are capable of handling simple electronic equipment and can benefit from it in terms of more confidence and enhanced positive effect [2].

The aging of the population necessitates better and more effective health care systems and technologies. Thus, picturing this scenario, a target can be identified in smoothing and delaying the transition that leads the person in his old age from the independent to the assisted living. Aging population will inevitably change society and elderly living dynamics. Optimization of resources, independent living and enhancement of the social, working, and physical activities of the elderly are the key aspects of this change.

The current paradigm to address this need is Ambient Assisted Living (AAL), where elderly persons are supported in their ability to live independent and high-quality lives by empowering the environment around them. Such environments have provided the ability to equip the home environment with a layer of technology to provide a truly 'Smart Home'. These homes offer improved living conditions and levels of independence for the population who require support with both physical and cognitive functions. At the core of the Smart Home is a collection of sensing technology which is used to monitor the behaviour of the inhabitant and their interactions with the environment [3]. Home Automation can provide a two-way contribution: it represents an opportunity to help overcoming difficulties; while its pervasive instrumentation provides precious information. Being aware of the elderly's activities has several applications: for the families, reassuring them about their beloved's safety; for caregivers, enabling them to provide prompt interventions.

Using the latest technology, AAL technologies aim to provide PwDs with the means to actively live their daily lives, protect their dignity, feel safe, maintain their capacities, sustain their integration with their communities, and help their caregivers in monitoring and preventing avoidable complications in consecutive treatment.

Researchers are currently considering to employ Home Automation not only as a commodity but also as an empowering tool to provide the elderly with security, independence, communication, etc. Although it cannot be neglected that the critical issues of designing an assistive tool are significantly different from those arising when designing a commercial product. The results of these innovations are leading toward a kind of system whose purpose is not only to provide a significant aid but also to collect information and allow monitoring some aspects of the inhabitant's life. Furthermore, the reliability of the designed system has to be considered crucial, since a system failure can cause disastrous consequences, threatening the system purpose and mission.

When healthcare systems are taken into account, activity recognition might contribute to the development of new rehabilitation systems for elderly care or health/fitness monitoring. Such assistive technologies have started to be adopted by smart homes and healthcare applications in practice and have delivered promising results for improving the quality of care services for the aging and provision of responsive assistance in emergency situations [4].

Activity learning from sensor data is the ability to discover and model activities based on collected sensor data. The number of methods that are now able to handle complex situations is increasing, and with them we are witnessing advances in the application of the theoretical techniques to challenge real-world problems including health monitoring and home automation. In the past, theories about behavior and activities were formulated based on restricted information, but recently, the maturation of sensor technologies has made available the data needed for the automation of activity learning. Sensors have become smaller, cheaper, and more powerful. At the same time, advances in the areas of wireless networks, data processing,

and machine learning have shifted researcher focus from low-level data collection and transmission to high-level information gathering and inference. Due these advances, automated modeling and tracking of activities have become integral components of numerous solutions to real-world problems. For example, surveillance and security systems try to recognize and predict activities in order to address terrorist threats. Ambient-assisted living exploits activity discovery and monitoring to support independent living for individuals with cognitive and physical impairments. Activity-aware services have made possible ideas such as intelligent meeting rooms, automated homes, and personalized digital assistants from science fiction to everyday life.

ADLs recognition usually accomplish their task by analyzing samples from sensors which are physically deployed in the ambient surroundings or worn by people. The aim of this work is to define a framework able to recognize the activity a human is carrying out in a real-time, sensor-rich environment. Applications of this work include the promotion of healthier lifestyles (e.g., encouraging exercising [5, 6]), preventing harmful habits (e.g., tobacco use or unwholesome food [7, 8]), detecting anomalous behaviors (e.g., fall detection [9, 10]), or tracking conditions (e.g., worsened mobility due to aging or illness [11]). Another goal is to discuss the challenges that are faced when theoretical techniques are applied to real-world problems while demonstrating functional uses for the proposed system. We also provide and analyze a real use-case scenario in order to validate the model.

Chapter 2

Related Work

Due to its many-faceted nature, it is necessary to differ and classify each type of AR so that the best approach may be pursued in each detailed scenario. In this chapter, AR is formally modelled, then the necessary devices and classic tools used to perform it are analyzed and we finally discuss the current state-of-the-art from the deep learning perspective.

2.1 Sensor-based activity recognition

Sensor-based activity recognition integrates the emerging area of sensor networks with novel data mining and machine learning techniques to model a wide range of human activities.

HAR aims to understand human behaviors in order to proactively assist users based on their requirement. Formally speaking, suppose a user is performing some kinds of activity belonging to a predefined activity set A:

$$A = \{A_i\}_{i=1}^{m}, \tag{2.1}$$

where m denotes the number of activity types. Considering there is a se-

quence of sensor that captures information:

$$s = \{d_1, d_2, \ldots, d_t, \ldots, d_n\}, \tag{2.2}$$

where d_t denotes the sensor reading at time t. We need to build a model to predict the activity sequence based on sensor reading s:

$$\hat{A} = \{\hat{A}_j\}_{j=1}^{n} = \mathcal{F}(s), \quad \hat{A}_j \in A, \tag{2.3}$$

while the true activity sequence (ground truth) is denoted as:

$$A^* = \{A_j^*\}_{j=1}^{n}, \quad A_j^* \in A \tag{2.4}$$

The goal of HAR is to learn a classifier by minimizing the discrepancy between predicted activities $\hat{A}$ and the ground truth A^*. Typically, a positive loss function $\mathcal{L}(\mathcal{F}(s), \hat{A})$ is constructed to reflect their discrepancy.

2.1.1 Types of activity

The terms 'action' and 'activity' are frequently used interchangeably in activity learning research. In fact, there is a tremendous diversity of concepts classified as activities in the literature. We refer to action as a simple ambulatory behavior executed by a single person and typically lasting short durations of time. Similarly, we refer to interaction as a short, single-movement action that involves multiple individuals. By contrast, we refer to activities as complex behaviors consisting of a sequence of actions that can be performed by a single individual or several individuals interacting with each other. Note that these definitions are hierarchical. An action may consist of a sequence of states, while activity may contain any number of actions as shown in Figure 2.1 [12].

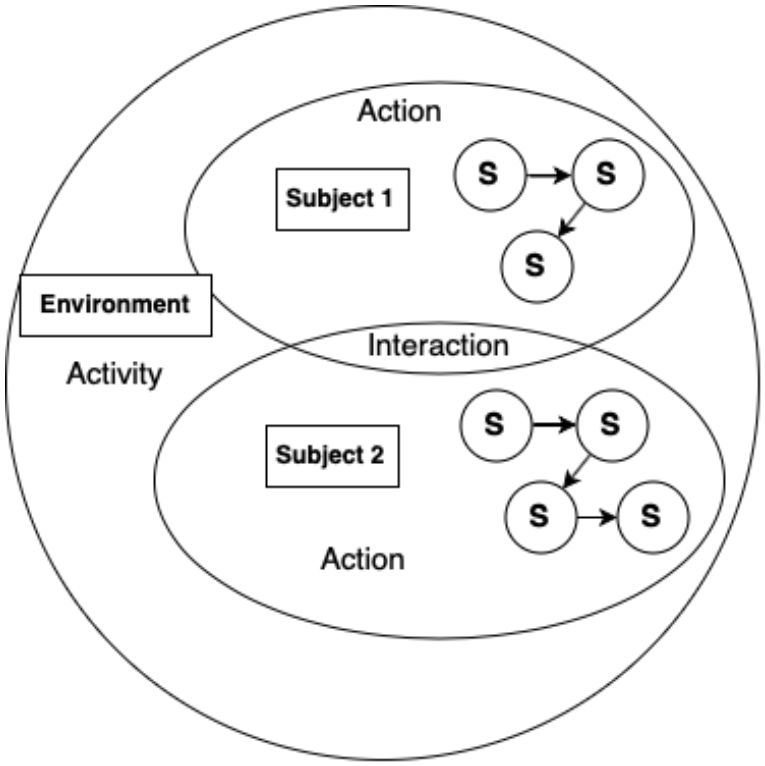

Figure 2.1: Relationship between state (S), action and activity.

2.1.2 Sensors for Activity Recognition

Sensors are devices that detect and respond to signals. A sensor converts a physical parameter such as temperature, blood pressure, or humidity into a signal that can be measured electrically and reported as a sensor event. A wide variety of sensors are available for activity learning and monitoring. In recent years, these sensors have also become low cost, wireless, and deployable in real-world, mobile settings. As a result, sensors are ubiquitous and are being integrated into more diverse computational settings. In order to design a sensor-based approach to activity learning, we need to consider: the types of sensors that are being used, how they are positioned in the environment [13] and how the data can be preprocessed for integration into data analysis and machine learning algorithms.

In the following sections we review two classes of sensors that are commonly used for activity learning and describe the type of data they generate.

Sensors in the Environment

These type of sensors are not fixed to the individuals but are placed in the environment surrounding the users. These are particularly valuable because they provide information without requiring individuals to comply with rules regarding wearing or carrying sensors in prescribed manners. This is a notable aspect because, in real-systems, we need to consider other than the functionality, the invasiveness of the solution adopted. Even though environment sensors can monitor activities for a group of individuals, they may have difficulty separating movements or actions among individuals that are part of that group.

Here we describe sensors that are commonly found installed in physical environments:

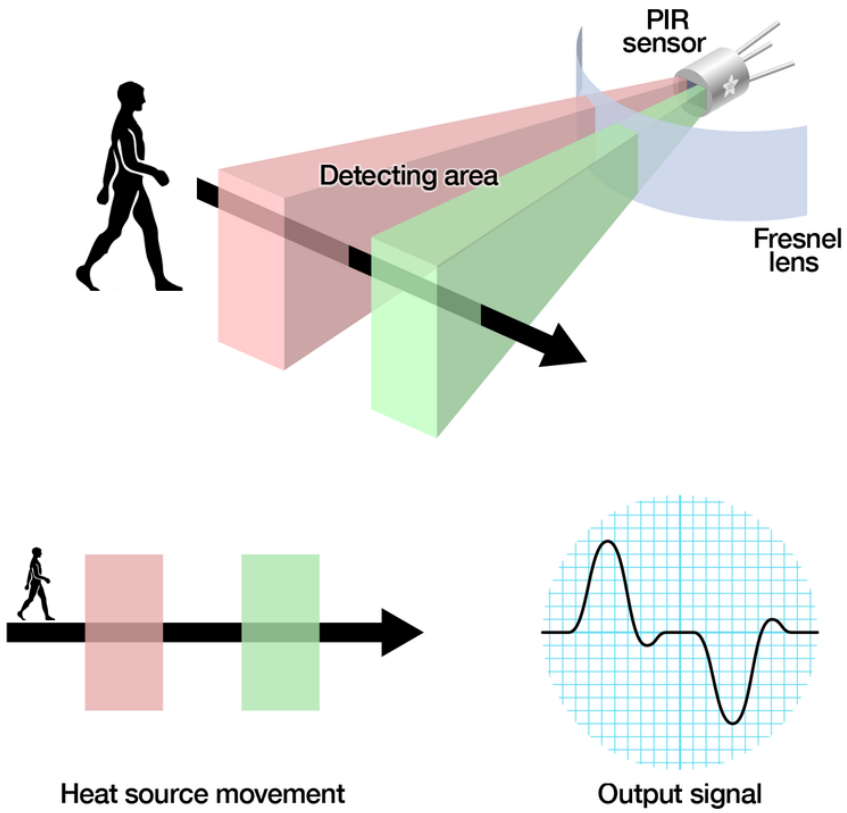

Figure 2.2: PIR sensor operating with an individual crossing the detection area.

Passive Infrared (PIR) Sensor: also referred to as motion sensors, they measure infrared (IR) light radiating from objects in its field of view. If the difference in the detected radiation between the multiple slots of a PIR sensor is greater than a predefined threshold (as would happen when a warm body moves into or out of the range of the sensor), it generates a message. Humans emit infrared radiation, so PIR sensors can be used to detect their movement within their coverage area. Because a passive infrared sensor (PIR) is sensitive to heat-based movement, it can operate in dim lighting conditions. On the other hand, it may not sense movement if an object is between the sensor and the moving person. All objects with a temperature above absolute zero emit heat energy in the form of radiation, so this kind of sensors may generate messages even if a device like a printer starts printing a large document in the vicinity. Figure 2.2 illustrates a PIR sensor scenario.

Magnetic Door Sensor: These devices are made up of two parts that form a circuit when they are kept parallel to each other. When someone opens the door (or any kind of openable section), the two parts separate and break the circuit, causing a change in the state of the sensor that can be reported as a sensor event. The same way when the door is closed back, the circuit is complete again and an event occurs. This sensor is useful for detecting if doors, windows, drawers, or cabinets are open or shut.

Temperature Sensor, Light Sensor, Humidity Sensor: Additional sensors can be placed in environments to measure ambient temperature, lighting, and humidity. Such sensors can be programmed to

periodically report their current status or when their current reading is sufficiently different from the previous time point.

Pressure Sensors: A pressure sensor is a device for pressure measurement of any gases or liquids. Pressure is an expression of the force required to stop a fluid from expanding and is usually stated in terms of force per unit area. For monitoring of activities in a particular environment, tactile sensors are sensitive to touch or force. These pressure devices detect and measure the interaction between an individual and a contact surface, they can be placed, under or over, door mats, chairs, beds, and floors. They are used to monitor the location and weight distribution of an individual in the designed surface.

Global Positioning System (GPS) Sensors: GPS is a satellite navigation system used to determine the ground position of an object. These devices rely on communication with at least four global positioning system (GPS) satellites and thus are not usable in every setting. Moreover, since GPS receivers require a relatively unobstructed path to space, GPS technology is not ideal for indoor use and is common to opt for different systems. Some examples being Dead reckoning or BLE Beacons.

BLE Beacons: A beacon is a very small device that contains a power source, a chipset, and an antenna to communicate with bluetooth equipped devices up to 230 feet away. Beacons cannot use Bluetooth, because it would use too much power. Instead, beacons are enabled by a technology called "Bluetooth Low Energy" (BLE) which has such low energy requirements that it can last up to 3 years on a single coin cell battery. Bluetooth beacons transmit a universally unique identifier

Figure 2.3: An assortment of Beacons devices from different vendors

picked up by a compatible app or operating system. The identifier and several bytes sent with it can be used to determine the device's physical location, track customers, or trigger a location-based action on the device such as a check-in on social media or a push notification.

Bluetooth beacon transmitters come in a variety of forms, including small coin cell devices and USB sticks. In AR they are used as indoor positioning system. Figure 2.3 illustrates a variety of Bluetooth beacon technologies.

Sensors on the Body

In addition to placing passive sensors around the environment, the individual can also wear, or carry, additional sensors that collect data about their gestures, movement, location, interactions, and actions. Wearable devices are smart electronic gadgets (electronic device with micro-controllers) that can be incorporated into clothing or worn on the body as implants or accessories.

Many of these sensors are now directly embedded on smartphones that are routinely carried by individuals as they perform their daily activities diminishing their invasiveness on the user and enabling the development of

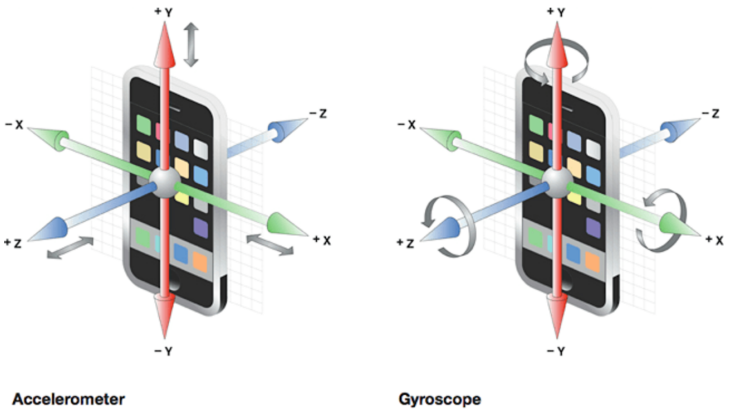

Figure 2.4: Triaxial acelerometer and gyro on a mobile phone.

new systems.

Gyroscope: Gyroscopes, or gyros, are devices that measure or maintain rotational motion. MEMS (microelectromechanical system) gyros are small, inexpensive sensors that measure angular velocity. The units of angular velocity are measured in degrees per second ($^{\circ}/s$) or revolutions per second (RPS). Angular velocity is simply a measurement of the speed of rotation and is measured along the x, y and z-axes.

Accelerometer: An accelerometer is an electromechanical device that can measure acceleration forces. These forces may be static, like the constant force of gravity pulling at your feet, or they could be dynamic, caused by moving or vibrating the accelerometer. This type of sensors can be worn or carried by an individual. Acceleration changes can indicate the beginning or ending of a motion. This three-dimensional sensor measures acceleration along the x, y, and z-axes. Acceleration is calculated as the change in velocity over time, or $\alpha = \triangle V / \triangle t$. Figure 2.4 illustrates accelerometer and gyroscopic smartphone sensors.

Magnetometer: This sensor measures the strength of the magnetic field in three dimensions. A magnetometer is valuable for providing some sensor orientation and for detecting and locating metallic objects within its sensing radius. Human activities can then be modeled based on their proximity to these detected objects.

2.2 Machine Learning in AR

Many machine learning algorithms have been designed, enhanced, and adapted for activity learning. The types of machine learning models that have been used for activity learning vary almost as greatly as the types of sensors from which data is collected.

In this section, is provided a brief overview of the basic supervised machine learning algorithms, diving deeper in recent approaches that belong to the field of Deep Learning.

2.2.1 Classic Approaches

With Supervised algorithm, we describe a branch of machine learning processes that learn a mapping from the input data to another value, which is usually a label that identifies a class. In fact, the algorithm extracts attributes from the input, also called features and tries to learn a function that returns the right label, given the features. The training process consists in minimizing a function defined as the difference between the return value of the mapping and the correct label of the input. We refer to this as Loss minimization.

The discovered relationship is represented in a structure known as the model. Models characterize the hidden relationship between the input data

and target attribute; they can be used for predicting the target attribute (label), given the values of the input data. This kind of prediction is called classification.

Classification models can be divided in two categories: *generative* and *discriminative*. Generative models learn the underlying probabilistic activity model that generates the sensor sequences. Examples of these approaches include naive Bayes Classifiers [14], Gaussian Mixture models (GMM) [15] and Hidden Markov models (HMM) [16, 17]. On the other hand, discriminative models focus on characterizing the region in the feature space that separates the different activity classes. Approaches in this category include support vector machines (SVMs) [18], decision trees [19], conditional random fields (CRFs) [20], and meta-learners such as boosting and bagging [21].

2.2.2 Benefits of Neural Network Modeling

Traditionally, methods from the field of signal processing were used to analyze and refine the collected sensor data.

Such methods were for feature engineering, creating domain-specific, sensor-specific, or signal processing-specific features and views of the original data. Statistical and machine learning models, like those mentioned in the previous section, were then trained on the processed version of the data [22].

A limitation of these approaches is that they may heavily rely on heuristic handcrafted feature extraction, which is usually limited by human domain knowledge. Furthermore, only shallow features can be learned by those approaches, leading to undermined performance. Moreover, the expertise would be required for each new dataset or sensor modality. In essence, it is expensive and not scalable.

Ideally, learning methods that can automatically learn the features re-

quired to make accurate predictions from the raw data would allow new problems, new datasets, and new sensor modalities to be adopted quickly and cheaply.

Recently, deep neural network models have started delivering state-of-the-art results for human activity recognition. They are capable of performing automatic feature learning from the raw sensor data and out-perform models fit on hand-crafted domain-specific features.

There are two main approaches to neural networks that are appropriate for time series classification and that have been demonstrated to perform well on activity recognition using sensor data.

They are Convolutional Neural Network Models and Recurrent Neural Network Models. In fact, RNN and LSTM are recommended to recognize short activities that have natural order while CNN is better at inferring long term repetitive activities. The reason is that RNN could make use of the time-order relationship between sensor readings, and CNN is more capable of learning deep features contained in recursive patterns [22].

CNN and LSTM

Both types of neural networks are suitable for time series classification but require that data is prepared in a specific manner in order to fit a model. That is, in a ‘supervised learning‘ form that allows the model to associate signal data with an activity class.

A straight-forward data preparation approach that was used both for classical machine learning methods on the hand-crafted features and for neural networks involves dividing the input signal data into windows of signals, where each window may represent a number of seconds of observation data. This is often called a *‘sliding window’*.

It is convenient to describe the shape of our prepared sensor data in terms of the number of samples, the number of time steps in a window, and the number of features observed at each time step:

$$[samples, timesteps, features] \tag{2.5}$$

There is no best window size, and it really depends on the specific model being used, the nature of the sensor data that was collected, and the activities that are being classified.

There is a tension in the size of the window and the size of the model. Larger windows require large models that are slower to train, whereas smaller windows require smaller models that are much easier to fit [23].

There is some risk that the splitting of the stream of sensor data into windows may result in windows that miss the transition of one activity to another. As such, it is traditionally common to split data into windows with an overlap such that the first half of the window contained the observations from the last half of the previous window ($overlap = 50\%$) [24].

In the adoption of neural network models, the use of overlaps, such as a 50% overlap, will double the size of the training data, which may aid in modelling smaller datasets, but may also lead to models that overfit the training dataset.

Convolutional Neural Network Models

Convolutional Neural Network models, or CNNs for short, are a type of deep neural network that was developed for use with image data, e.g. such as handwriting recognition [25].

They have proven very effective in challenging computer vision problems for tasks such as identifying and localizing objects in images and automati-

cally describing the content of images. They are models that are comprised of two main types of elements: *convolutional layers* and *pooling layers*.

Convolutional layers read an input, such as a 2D image or a 1D signal, using a kernel that processes small segments at a time and steps across the entire input field. Each read results in a convolution between the input and a filter map and represents an internal interpretation of the input.

Pooling layers take the feature map activations and distil them to the most essential elements, such as using a signal averaging or maximizing process. The convolution and pooling layers can be repeated at depth, providing multiple layers of abstraction of the input signals. The output of these networks is often one or more fully connected layers that interpret what has been read and map this internal representation to a class value.

When applied to time series classification like HAR, CNN has two advantages over other models: *local dependency* and *scale invariance*. Local dependency means the nearby signals in HAR are likely to be correlated, while scale invariance refers to the scale-invariant for different paces or frequencies [22].

The first important work using CNNs to HAR was by Ming Zeng, et al [26]. In the paper, the authors develop a simple CNN model for accelerometer data, where each axis of the sensor is fed into separate convolutional layers, pooling layers and then concatenated before being interpreted by hidden fully connected layers.

Figure 2.5 clearly shows the topology of the model. It provides a good template for how the CNN may be used for HAR problems and time series classification in general.

Quite large CNN models were developed, which in turn allowed the authors to claim state-of-the-art results on challenging standard human activity

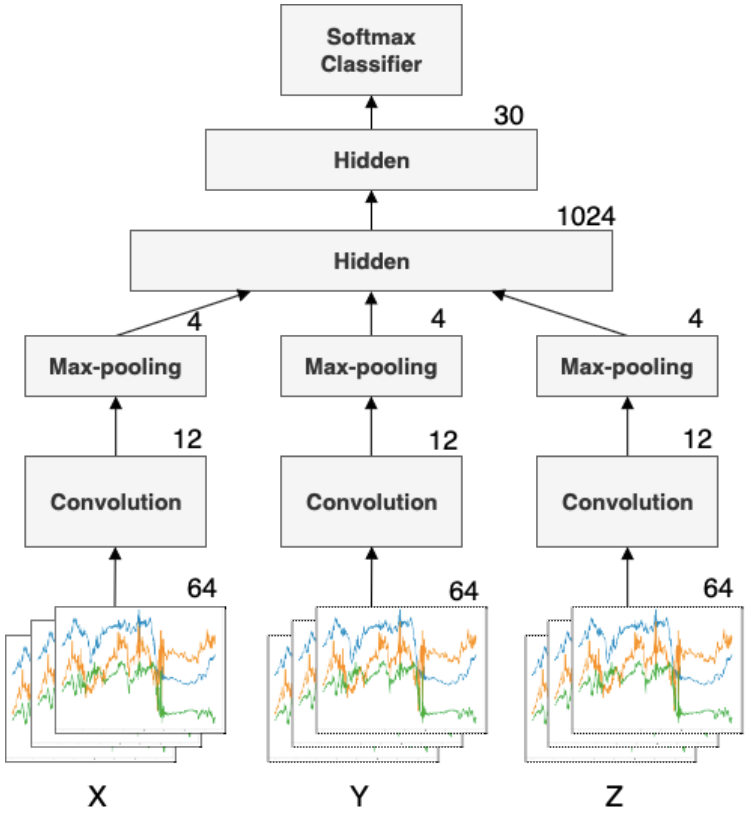

Figure 2.5: Depiction of CNN Model for Accelerometer Data. Each axis of the sensor is fed into separate convolutional layers, pooling layers and then concatenated before being interpreted by hidden fully connected layers [26].

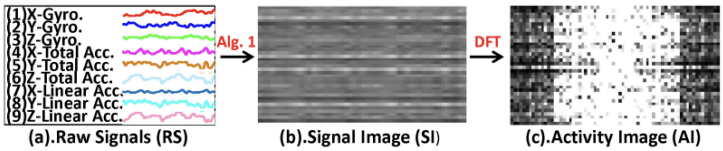

Figure 2.6: Processing of Raw Sensor Data into an Image. The signals are combined to produce an image on which is applied DFT. Then the sample is fed to a 2D CNN [28].

recognition datasets. For example, 3D-CNN architecture where proved more suitable on video for AR [27].

Another interesting approach was proposed where instead of using 1D CNNs on the signal data, they combine the signal data together to create "images" which are then fed to a 2D CNN and processed as image data with convolutions along the time axis of signals and across signal variables, specifically accelerometer and gyroscope data [28]. Figure 2.6 illustrates the process.

Finally, another work on the topic shows that convnets indeed derive relevant and more complex features with every additional layer, although the difference of feature complexity level decreases with every additional layer. A wider time span of temporal local correlation can be exploited (1×9 – 1×14) and a low pooling size (1×2 – 1×3) is shown to be beneficial [29].

Recurrent Neural Network Models

Recurrent neural networks, or RNNs for short, are a type of neural network that was designed to learn from sequence data, such as sequences of observations over time, or a sequence of words in a sentence.

A specific type of RNN called Long Short-Term Memory network, or LSTM for short, is perhaps the most widely used RNN as its careful de-

sign overcomes the general difficulties in training a stable RNN on sequence data [30].

LSTMs have proven effective on challenging sequence prediction problems when trained at scale for such tasks as handwriting recognition [31], language modeling [32], and machine translation [33].

A layer in an LSTM model is comprised of special units that have gates that govern input, output, and recurrent connections, the weights of which are learned. Each LSTM unit also has internal memory or state that is accumulated as an input sequence is read and can be used by the network as a type of local variable or memory register.

Like the CNN that can read across an input sequence, the LSTM reads a sequence of input observations and develops its own internal representation of the input sequence. Unlike the CNN, the LSTM is trained in a way that pays specific attention to observations made and prediction errors made over the time steps in the input sequence, called backpropagation through time.

The LSTM learns to map each window of sensor data to an activity, where the observations in the input sequence are read one at a time, where each time step may be comprised of one or more variables (e.g. parallel sequences).

One example is by Abdulmajid Murad and Jae-Young Pyun [34]. In their paper, they comment on the limitation of CNNs in their requirement to operate on fixed-sized windows of sensor data, a limitation that LSTMs do not strictly have. They explore the use of LSTMs that both process the sequence data forward and in both directions (*Bidirectional* LSTM). Interestingly, the LSTM predicts an activity for each input time step of a subsequence of sensor data, which are then aggregated in order to predict an activity for the window.

Moreover, it may be useful to use an LSTM in conjunction with a CNN

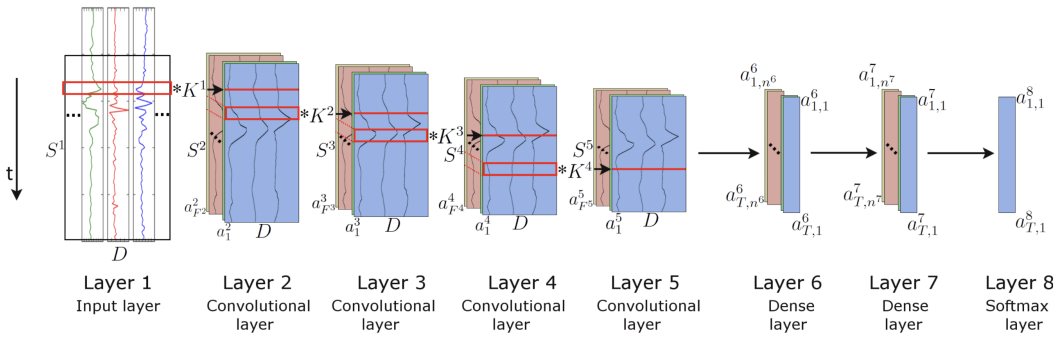

Figure 2.7: Depiction of CNN LSTM Model for Activity Recognition. CNN model is used to extract the features from raw sample data, and output features from the CNN are then interpreted by an LSTM [36].

on HAR problems, in a CNN-LSTM model or ConvLSTM model [35]. This is where a CNN model is used to extract the features from a subsequence of raw sample data, and output features from the CNN for each subsequence are then interpreted by an LSTM.

Another example is [36], where the authors claim that the removal of the pooling layers is a critical part of their model architecture. The use of pooling layers after the convolutional layers interferes with the convolutional layers' ability to learn to downsample the raw sensor data before the LSTM. Their archiecture is shown in Figure 2.7.

2.3 Contributions of this thesis with respect to the state-of-the-art

Taking advantage of the numerous works developed using deep learning in the field of AR, in this thesis, we propose a novel framework for activity Learning. The model will be able to extract features from multiple types of sensors located in a sensor-rich environment and make a prediction based on

their output. Both CNN and LSTM architectures will be used in a multi-input manner to enhance the classification accuracy making use of all the data sources. In addition, an average ensemble of the same model, but with different window sizes is used to overcome the choosing size problem discussed previously and to learn to discern activities of different complexity. In the next section is explained and discussed the framework in detail, exploring the unprecedented use of multi-input Neural Networks in AR and the *temporal ensemble*, motivating the choices of design and evaluating its ability to generalize in different scenarios.

Chapter 3

Model framework

Due to the rapid advances in sensing and computing technology, multiple sensors can now be used to simultaneously monitor humans inside their home. This creates a data-rich environment, enabling an unprecedented opportunity to reach a better understanding and make inference about the current and future behaviour of users in real time.

Depending on specific health tasks, a human should perform certain activities in their daily life, for example, individuals at risk that must take a medication every day or someone with an obesity problem that has to perform physical activities. AR can help to monitor the behaviour of the individual, checking if he is carrying out a healthy lifestyle.

Multisensor fusion refers to the synergistic combination of sensory data from multiple sensors to provide more reliable and accurate information. The integration or fusion of redundant information can reduce overall uncertainty and thus serve to increase the accuracy with which the information is perceived by the system [37]. Multiple sensors providing redundant data can also serve to increase reliability in the case of sensor error or failure. For example, if there are multiple wearable accelerometers that are gathering movement

data related to activities, multisensor fusion aids in scenarios when some of the accelerometers fail or provide noisy data.

Complementary information from multiple sensors allows features in the environment to be perceived that are difficult or impossible to perceive using just the information from each individual sensor operating separately [38]. For example suppose that, in a smart home setting, multiple sensors such as motion, temperature, and pressure sensors are gathering complementary data about the *Cooking* activity. Motion sensors can provide data about a human presence in the kitchen area, temperature sensors provide clues to whether the stove is on and pressure or vibration sensors can indicate whether any kitchen object is being used. While these three sensor classes may independently be weak at characterizing the *Cooking* activity, fusing them together leads to a stronger model.

In this chapter we describe the design of the framework proposed for HAR in a smart home, that is, an environment characterised by the presence of multiple sensors for activity recognition. The model in this section is formulated to work with different types and amounts of sensors. The chapter goes through the modeling of the problem, the data preparation and the design of the architecture. For a more detailed, case specific explanation, refer to Chapter 4.

3.1 Data Fusion

In a typical scenario for AR, each sensor collects information independently as a stream of data. The data is then passed to a hub that gathers all the information coming from the sensors. With the aim of fusing the data together, we suggest to divide sensors in two categories: *continuos sensors*,

like accelerometers, gyros or magnetometer, that produce output constantly at some fixed rate, and *event-based sensors*, like motion or binary sensors, that emit an output every time there is some kind of change in the physical parameter they are monitoring.

Let C be the stream coming from a set of *continuous sensors*, and E an *event-based* sensors' stream:

$$C_{t_1,t_n} = \{a_{t_1}, a_{t_2}, \ldots, a_{t_n}\} \text{ where } a_t =< id, v >, \tag{3.1}$$

where a_t is a tuple representing the value v collected from the sensor with identification number id at the timestamp t.

Because of the nature of C, we know that:

$$t_{i+1} - t_i = t_{j+1} - t_j \quad \forall i, j. \tag{3.2}$$

If we also know the sampling frequency fc of the sensor, we can derive a useful relationship:

$$K_{i,j} = fc \times \triangle(t_i, t_j), \tag{3.3}$$

where $K_{i,j}$ is the amount of samples collected by the sensor in a time range defined by t_i and t_j. The time elapsed $\triangle(t_i, t_j)$ with $t_j > t_i$ can be calculated as $t_j - t_i$.

We cannot,however, draw the same conclusion for E:

$$E_{t_1,t_n} = \{d_{t_1}, d_{t_2}, \ldots, d_{t_n}\} \text{ where } d_t =< id, v >, \tag{3.4}$$

since equation 3.2 is not valid. We thus need to synchronise C_{t_i,t_j} and E_{t_m,t_n}, to obtain usable data for our model.

The first step is to perform alignment. That is, to find the set of timestamps defined as $\{t\}_C \cap \{t\}_E$ where $\{t\}_X$ is the set of timestamps recorded in the stream X. We make use of some high level utilities:

- *max(arg1, arg2)*: returns the maximum value between arg1 and arg2
- *min(arg1, arg2)*: returns the minimum value between arg1 and arg2
- *search(arg1, list)*: returns the index of the most similar value to *arg1* in *list*

The routine for stream alignment is presented considering as input just two streams for the sake of simplicity. This process is easily reproducible for a variable number of inputs:

- C, E: input streams timestamps.
- N, M: length of C and E.
- C', E' : output streams timestamps.

Algorithm 1 Stream *alignment* procedure.

1: **procedure** ALIGN(C, E, N, M)
2: $t\_latest\_start = max(C[0], E[0])$
3: $t\_earliest\_end = min(C[N], E[M])$
4: $new\_start\_C = search(t\_latest\_start, C)$
5: $new\_start\_E = search(t\_latest\_start, E)$
6: $new\_end\_C = search(t\_earliest\_end, C)$
7: $new\_end\_E = search(t\_earliest\_end, E)$
8: $C' = [new\_start\_C : new\_end\_C]$
9: $E' = [new\_start\_E : new\_end\_E]$
10: $E'[0] = C'[0]$
11: $E'[length(E')] = C'[length(C')]$

After alignment, the streams have the same starting and ending timestamps. We note that some kind of priority is given to the continuous stream

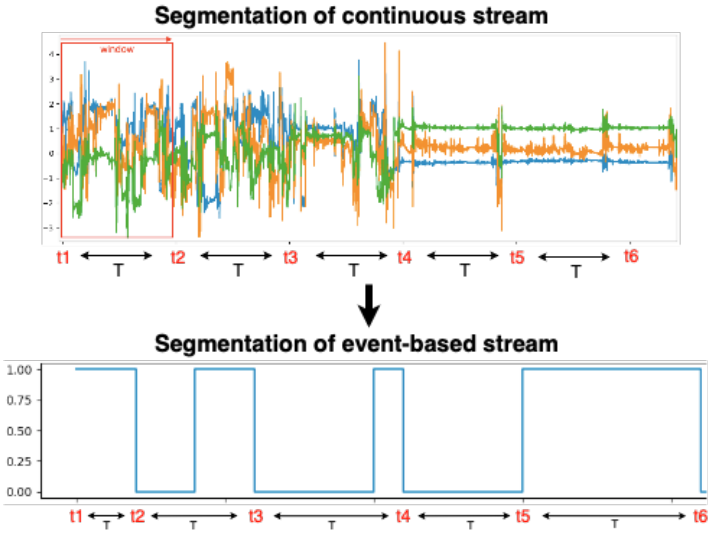

Figure 3.1: Windowing scheme fo the streams. The time ranges obtained segmenting the continuous streams are used to window the event-based streams.

C, this is because, generally, the continuous stream has a higher rate of transmission compared to the event-based. As can be seen in lines 10-11, we assign the first and last timestamp of C to E, because a change in the timestamp of C would affect the validity of the Equation 3.2 shown above.

3.2 Data Segmentation

Signal segmentation is a crucial stage in the activity recognition process and it consists in the streaming sensor events being segmented into chunks representing a fixed time range.

We can apply a sliding window on the *continuous* sensor stream to extract the timestamp range of each sample and use it to window the *event-based* stream. After deciding the length of time for the windows, is sufficient to apply equation 3.3 to derive the length in samples.

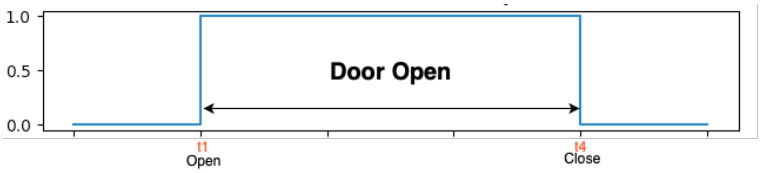

Figure 3.2: Magnetic Door sensor stream while the door is being used. Note that the only explicit information in the stream would be $Open_{t1}$ and $Closed_{t4}$.

Note how in Figure 3.1 the length of time T results in a fixed length in the upper stream while it amounts to a variable number of samples in the other one. For this reason, samples from the *event-based* streams needs to be additionally processed to be used as input for the model. It is helpful to divide this type of data in two sub-categories:

- *with-memory*: for this type of sensors, the events are affected by previous activations of the sensor or in some way two distinct events are needed to describe an action. As an example, we can think of Magnetic Door sensor as shown in Figure 3.2. If the Door stream is $D = \{Open_{t_1}, Closed_{t_4}\}$ this obviously means that in the time range $t_1 < t < t_4$, the door was still open and we must encode this information.

- *memory-less*: for this type of sensor, an event fully describes the state of the measured physical parameter. As an example, we can think of BLE Beacons, where at each timestamp, the nearest beacon is reported and there isn't a lasting duration for the action.

The processing needed to deal with this type of sensor will be explained in Chapter 4.

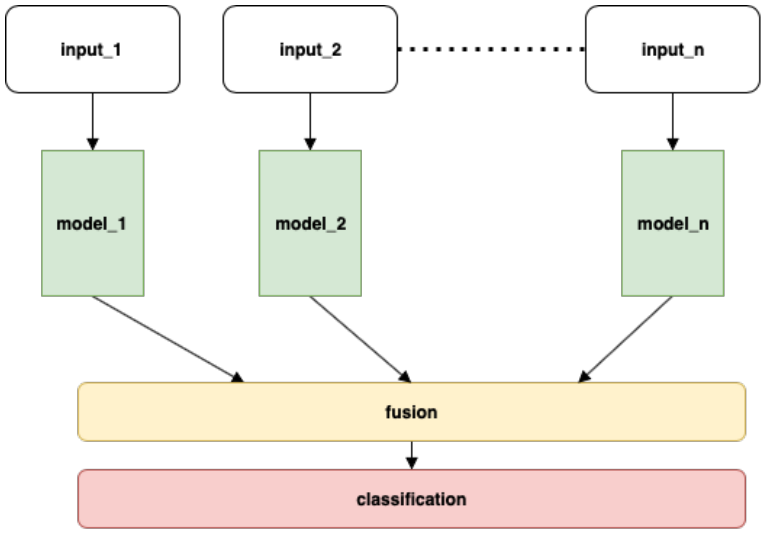

Figure 3.3: Multi-input neural network with n inputs. We adopt a late fusion strategy where features extracted from independent input modalities are fused via concatenation before the final classification layer.

3.2.1 Multi input neural network

After segmentation, the data format is:

$$[n\_samples, n\_channels, window\_length] \tag{3.5}$$

for each stream and is ready to be used as input for our model. We will apply a mid-level data fusion using a neural network, with so many inputs as there are different sources. Figure 3.3 shows the general model framework where a NN receives n inputs, one for each data source.

During the forward pass, each data source is kept separated until deeper in the network, at which point we concatenate the feature representations and then proceed with a single, merged processing pipeline in the network which ends at a single classification layer. We can think of it as independent pipelines that try to learn a *feature representation* for each source and then combine these representations to perform classification. This is particularly

useful because we are able to maintain the independence of each network branch so that we can design a different architecture defining the feature extraction process for each independent source. For example, we could have an LSTM processing data coming from an accelerometer and a CNN processing the activation of the motion sensors.

3.2.2 Temporal Ensemble

Because the size of the window T is a long-discussed problem, and it is difficult to find the best size in each scenario, in this work we make use of many models, each one trained to classify an activity using different values for T.

A sample of T seconds is classified using multiple models. Each model M_i makes a number of prediction calculated as T/T_i where $T_i < T$ is the expected length in seconds of samples for M_i. Predictions are averaged twice, once locally for each model, so that for each model we have a single prediction, and then globally, so that we have a single prediction from the ensemble.

The only suggestion or soft constraint for choosing the size in seconds of the inputs for the ensemble is that:

$$T \bmod T_i = 0, \qquad \forall i. \tag{3.6}$$

This guarantees that a sample of length T, can be divided obtaining a number of segments T/T_i. In Figure 3.4 presented below, we show an example.

Ensembles, in general, are likely to increase the model accuracy, but in the case of AR they are particularly useful because of the variability in the granularity of the activities that we want to classify, this is why we refer to it as a *temporal ensemble*. As an example we can think of the two distinct activities: *throwing litter into the bin* and *relaxing on the sofa*. The first one

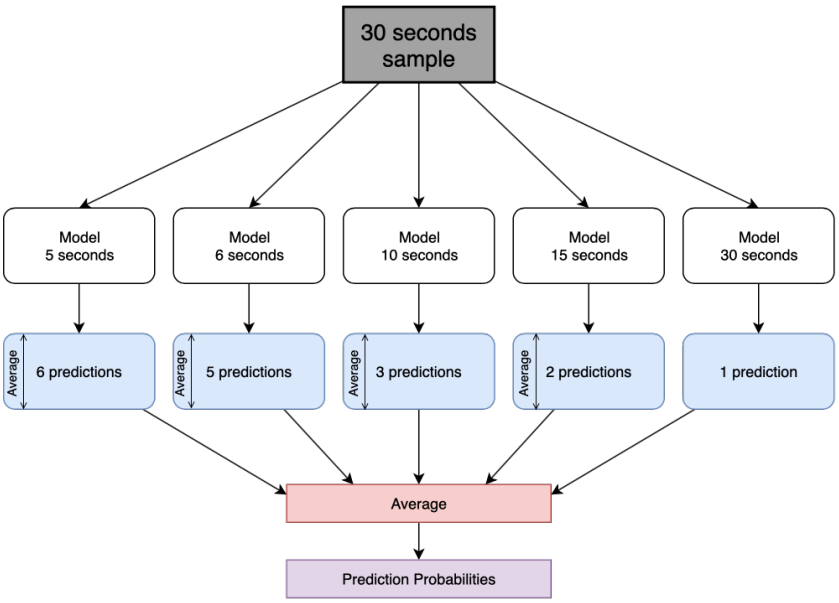

Figure 3.4: Average *(temporal)* ensemble example using T=5, 6, 10, 15, 30. We make use of many models, each one trained to classify an activity using different values for T. Predictions are averaged twice, once locally for each model, so that for each model we have a single prediction, and then globally, so that we have a single prediction from the ensemble.

is an instantaneous action so it's better represented in a short window, while the second is a longer constant activity, meaning it could be classified using higher values of T.

Chapter 4

Experimental Results

The UCAmI Cup is an event held within the context of the International Conference on Ubiquitous Computing and Ambient Intelligence [39], where participants are given the opportunity to use their tools and techniques to analyse a human activity recognition dataset and to compare their results with others working in the same domain. In the last conference, was presented a human activity recognition dataset related to activities of daily living generated in the UJAmI Smart Lab, University of Jaèn. The dataset represents 246 activities performed over a period of ten days carried out by a single inhabitant. The dataset includes four data sources:

- event streams from 30 binary sensors;
- intelligent floor location data;
- proximity data between a smartwatch worn by the inhabitant and 15 Bluetooth Low Energy beacons; and
- acceleration of the smartwatch.

UCAmI Cup aims to be an annual event within the forum of the International Conference of Ubiquitous Computing and Ambient Intelligence

(UCAmI) where delegates will be provided with the opportunity to use their tools and techniques to analyse a HAR dataset and to compare their results with others in the ADL context. In this chapter we present the dataset used to train and test the model, explaining in detail the design of the network.

4.1 UCAmI

The UJAmI SmartLab measures approximately 25 square meters. It is divided into five regions: entrance, kitchen, workplace, living room and a bedroom with an integrated bathroom.

A set of multiple and heterogeneous sensors have been deployed in different areas of the environment in order to capture human-environment interactions in addition to inhabitant behaviour. The dataset is divided into two sets:

- Part 1: Labeled training set with seven days of recordings that contains 169 instances; and
- Part 2: Unlabeled test set with three days of recordings that contains 77 instances.

The dataset includes 24 different types of activities as presented in Tables 4.1 and 4.2.

Labels are provided in a separate file for each day of recording and defines beginning and end time of each activity. This results in ranges of time where the user wasn't performing any activity. In the next section, we will explore in detail the sensors present in the data set.

ID	Activity	Description
Act01	Take medication	Inhabitant goes to the kitchen, takes some water, remove medication from a box and swallows the pills.
Act02	Prepare breakfast	Inhabitant goes to the kitchen, takes some products and can: *i*) make a cup of tea with kettle or *ii*) makes a hot chocolate drink with milk in the microwave. He does not sit down to eat.
Act03	Prepare lunch	Inhabitant goes to the kitchen, takes some products from the refrigerator and pantry. Then he can: *i*) prepare a plate of hot food on the fire or *ii*) warm up a pre-cooked dish in the microwave. He does not sit down to eat.
Act04	Prepare dinner	Inhabitant goes to the kitchen, takes some products from the refrigerator and pantry. Then he can: *i*) prepare a plate of hot food on the fire, or *ii*) heates a pre-cooked dish in the microwave. He does not sit down to eat.
Act05	Breakfast	Inhabitant goes to the dining room in the kitchen in the morning and sits down to eat. When he is finished, he places the utensils in the sink or in the dishwasher.
Act06	Lunch	Inhabitant goes to the dining room in the kitchen in the afternoon and sits down to eat. When he is finished, he places the utensils in the sink or in the dishwasher.
Act07	Dinner	Inhabitant goes to the dining room in the kitchen in the evening and sits down to eat. When he is finished, he places the utensils in the sink or in the dishwasher.
Act08	Eat a snack	Inhabitant goes to the kitchen to take some fruit or a snack and eats it in the kitchen or in the living room. This activity can imply that the utensils are placed in the sink or in the dishwasher.
Act09	Watch TV	Inhabitant goes to the living room, takes the remote control, then sits down on the sofa.
Act10	Enter the SmartLab	Inhabitant enters the SmartLab through the entrance at the main door.
Act11	Play a videogame	Inhabitant goes to the living room, takes the remote controls of the TV and XBOX, and sits on the sofa.
Act12	Relax on the sofa	Inhabitant goes to the living room, then sits on the sofa.

Table 4.1: Activity classes in the UJA dataset.

ID	Activity	Description
Act13	Leave the SmartLab	Inhabitant goes to the entrance, opens the main door and leaves the SmartLab closing the main door.
Act14	Visit in the SmartLab	Inhabitant goes to the entrance, opens the main door, chats with someone at the main door, and then closes the door.
Act15	Put waste in the bin	Inhabitant goes to the kitchen, picks up the waste, then leaves the SmartLab. Usually, the inhabitant comes back after around 2 minutes.
Act16	Wash hands	Inhabitant goes to the bathroom, opens the tap, lathers, rinses and dries his hands.
Act17	Brush teeth	Inhabitant goes to the bathroom and brushes his teeth opening/closing the tap.
Act18	Use the toilet	Inhabitant goes to the bathroom and uses the toilet, opening/closing the toilet lid and pulling the cistern.
Act19	Wash dishes	Inhabitant goes to the kitchen and places dirty dishes in the dishwasher, then puts them in their place.
Act20	Laundry	Inhabitant goes to the bedroom, picks up the laundry basket, puts clothes in the washing machine, then waits around 20 minutes and then takes the clothes out of the washing machine placing them in the bedroom closet.
Act21	Work at the table	Inhabitant goes to the workplace, sits down doing work, and finally, gets up.
Act22	Dressing	Inhabitant goes to the bedroom, puts dirty clothes in the laundry basket, opens the closet, puts on clean ones and then closes it.
Act23	Go to bed	Inhabitant goes to the bedroom, lies in bed and sleeps. This activity terminates once the inhabitant stays 1 minute in bed.
Act24	Wake up	This activity involved the inhabitant getting up and out of the bed.

Table 4.2: Activity classes in the UJA dataset (continued).

Binary Sensors

In the UJAmI SmartLab, a set of 30 binary sensors were deployed (see Figure 4.1). All of them transmit a binary value together with the timestamp. The set of binary sensors are categorized into the following three sensor types:

- Magnetic contact. These are applied on doors and objects that have a fixed place when they are not being used. For example, a TV remote control, medicine box, or bottle of water. When the value is "close", it means that the object is not being used.

- Motion. When motion is detected the sensor sends a value that represents movement. When the movement ceases, the sensor sends a value that represents no movement.

- Pressure. This is a wireless sensor connected to a textile layer. When pressure is detected, the sensor sends a value that represents pressure. When the press ceases, the sensor sends a value that represents no pressure. This sensor is used in sofas, chairs or beds.

Proximity Sensors

The proximity data was collected through an Android application installed on the smartwatch of the inhabitant and a set of 15 BLE beacons (see Figure 4.2). When the smartwatch reads the signal from a BLE beacon, it collects a Received Signal Strength Indicator (RSSI) measurement. The greater the RSSI received by the smartwatch, the smaller the distance between it and the BLE beacon.

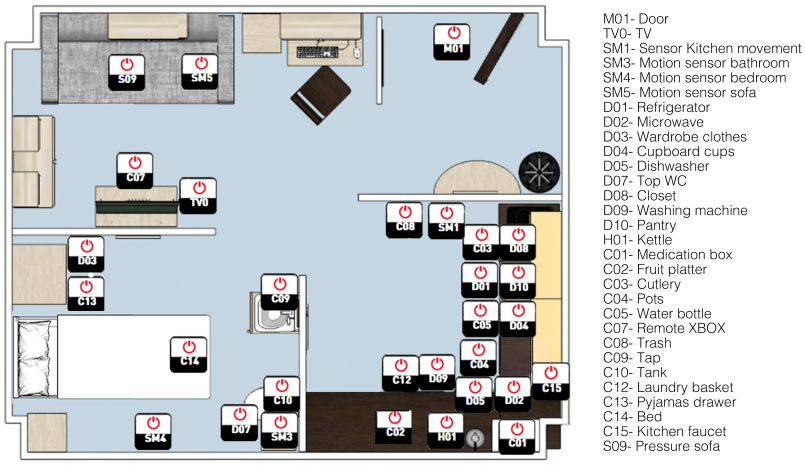

Figure 4.1: Layout of the binary sensors in the UJAmI SmartLab.

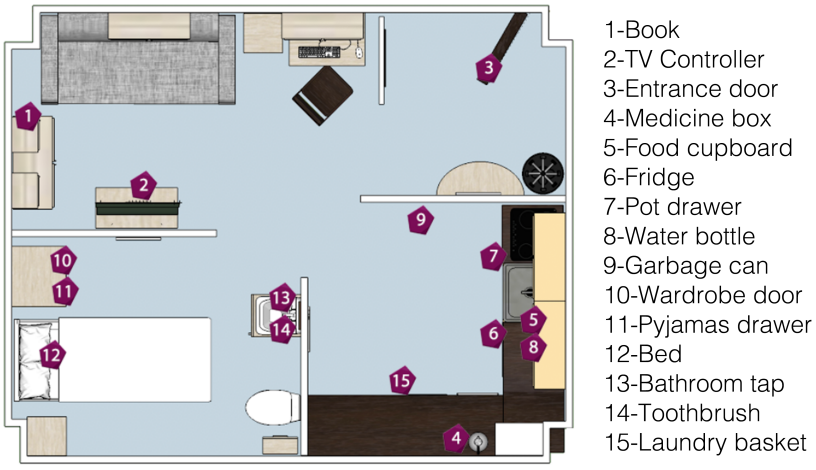

Figure 4.2: Layout of the proximity BLE Beacons in the UJAmI SmartLab.

Figure 4.3: Layout of the smart floor in the UJAmI SmartLab.

Acceleration Sensors

The acceleration data has been collected through an Android application installed on the smartwatch of the inhabitant. Data was collected with a sampling frequency of 50 Hz. The acceleration data has been collected in three axes, which are expressed by meter per second squared (m/s^2).

Floor pressure Sensors

The UJAmI SmartLab has a SensFloor that consists of a suite of capacitive sensor that lie below the floor. The floor of the UJAmI SmartLab is formed by 40 modules that are distributed in a matrix of 4 rows and 10 columns. A module is composed of eight sensor fields, each sensor in a module is associated with an id-number. The layout of the SensFloor in the UJAmI SmartLab is presented in Figure 4.3.

4.1.1 Data Processing

We now explain in detail the processing needed to feed each data source to the network and subsequently make predictions. As we described in Chapter 3, data streams in AR can be organized in continuous and event-based data. We include the accelerometer in the first group, and the rest of the sensor is classified as event-based.

We start by labelling the continuous stream by finding the intersection between the label file and the accelerometer stream. This process results in each sample having an id between 0 and 24 representing the activity that was carried out, where id=0 means that the user wasn't doing any activity. Then we proceed aligning the sources as described in Chapter 1, this is necessary because in the dataset the streams do not start or end at the same time.

Next, we must segment and encode each stream:

- After normalizing the accelerometer data, that is subtracting the mean and dividing by *standard deviation* each channel ($\frac{X-\mu}{\sigma}$), we apply a sliding window of some length T *seconds*. We start segmenting the accelerometer stream beacuse from it we can retrieve *start* and *end* timestamps of each sample in order to use them for the other sources:

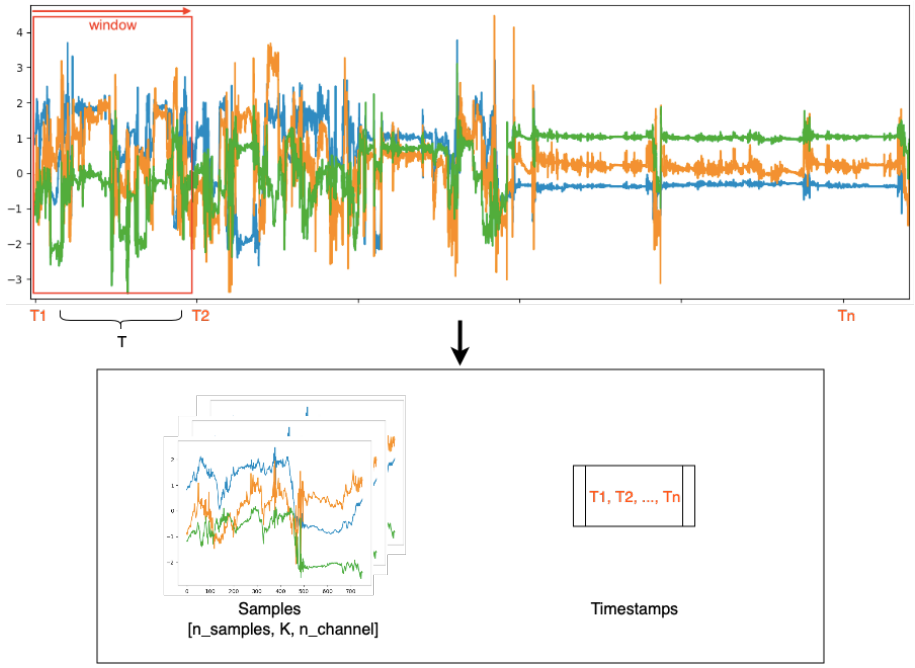

Figure 4.4: Sliding window applied to the accelerometer stream. By applying segmentation on the accelerometer stream we obtain the segmented version of the data and the timestamps vector. We then use the vector to segment the other sources.

We decided to apply 50% overlapping windows to increment the number of samples available for training. After segmentation, we obtain $n\_samples$ of dimension $[K \times n\_channel]$ where K is the length in samples of the window and can be calculated using Eq. 3.3 and in this case $n\_channel=3$, the channels of the tri-axial accelerometer. At this point, we can assign a label to each segment calculated as the most frequent class along the samples of that segment.

- Then we have to apply a sort of oversampling on the *with-memory* streams, i.e. binary sensor's data. From the stream, where each event e is collected at some timestamp t_k, we want to obtain a continuous

stream at the desired sample rate σHz, where two adjacent events are separated by a constant amount of time. This is done interpolating each couple of adjacent events (e_i, e_j), adding an amount of samples between them calculated as $(t_{e_j} - t_{e_i}) \times \sigma$. To decide the values of this artificial crafted events, we rely on the last value sent from that particular sensor.

We decided σ=*1Hz* so that we could monitor the state of all the sensors each second, facilitating the synchronization operation.

Suppose we have two binary sensors e and c generating the stream E:

$$E = \{e_{12:03:02}(0), c_{12:03:04}(1), c_{12:03:10}(0), e_{12:03:11}(1)\} \tag{4.1}$$

where $x_t(v)$ is the event from a sensor x with timestamp t, carrying the value v. The interpolation would generate E' as:

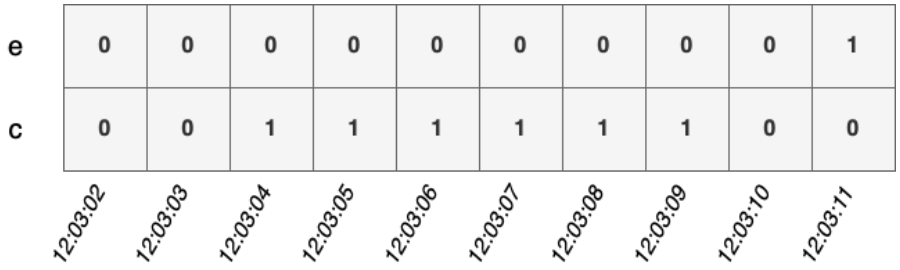

Figure 4.5: Interpolation process applied on the stream $E = \{e_{12:03:02}(0), c_{12:03:04}(1), c_{12:03:10}(0), e_{12:03:11}(1)\}$. We obtain a continuous stream at the desired sample rate (1 sample per second in this case).

To enhance the underlying information of the stream, we additionally encode the position of the sensors inside the smart home. This is possible applying an imaginary grid over the smart home plan, as shown in Figure 4.6, to map the layout as a matrix.

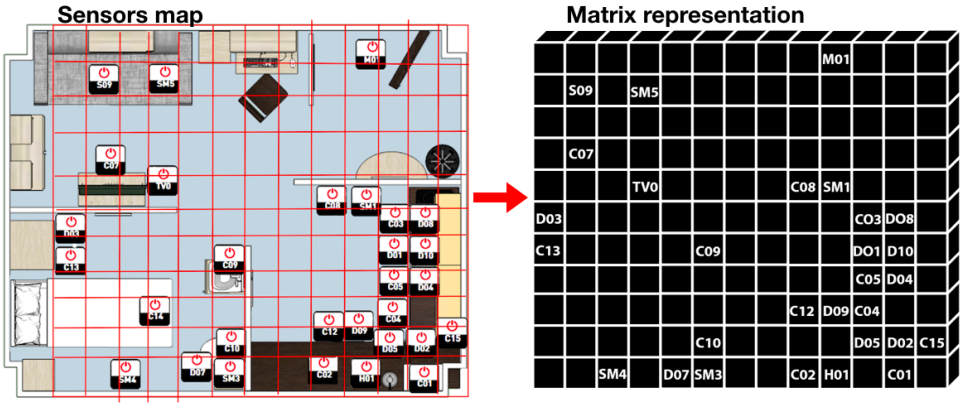

Figure 4.6: Mapping of the binary sensor using the layout of the smart-home. We obtain a 11×13 matrix that encodes the state of the sensors.

After mapping, we obtain a 3D structure representing the data stream. We use activity images converted from the logs to represent the data. These images have a black background and white pixels corresponding to regions of time where the sensors were sending positive signals ('Open', 'Movement', 'Pressure').

As an example we show in Figure 4.7 the complete process for a section of data taken from the dataset.

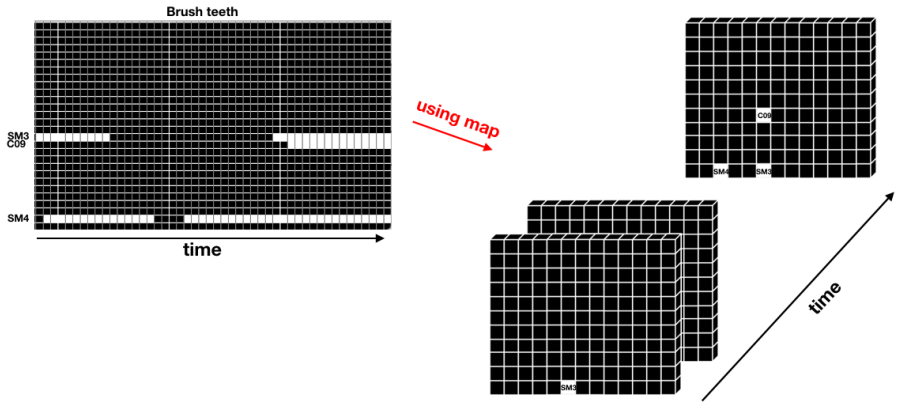

Figure 4.7: Mapping of a chunk of "brush teeth activity" into matrix form. Frome the left, raw binary data is processed by applying interpolation and then mapped using the layout. We obtain a sequence of frames over time that describes the evolution of the binary sensors.

After applying segmentation using the timestamps retrieved from the accelerometer, we obtain $n\_samples$ of dimension $[T \times n\_rows \times n\_columns]$ where T is the length in seconds of the window and n_rows, $n\_columns$ are the dimensions of the matrix used to grid the layout of the sensors, in our case $(11, 13)$.

- To process the proximity sensor, firstly we segment the stream using accelerometer's timestamps and then we apply one-hot-encoding, finally obtaining a vector of dimension $n\_sensors$ (15 in our case), representing the beacons in the vicinity of the user in each window of time. At the end of the process, we obtain $n\_samples$ arrays of length 15 as shown in Figure 4.8.

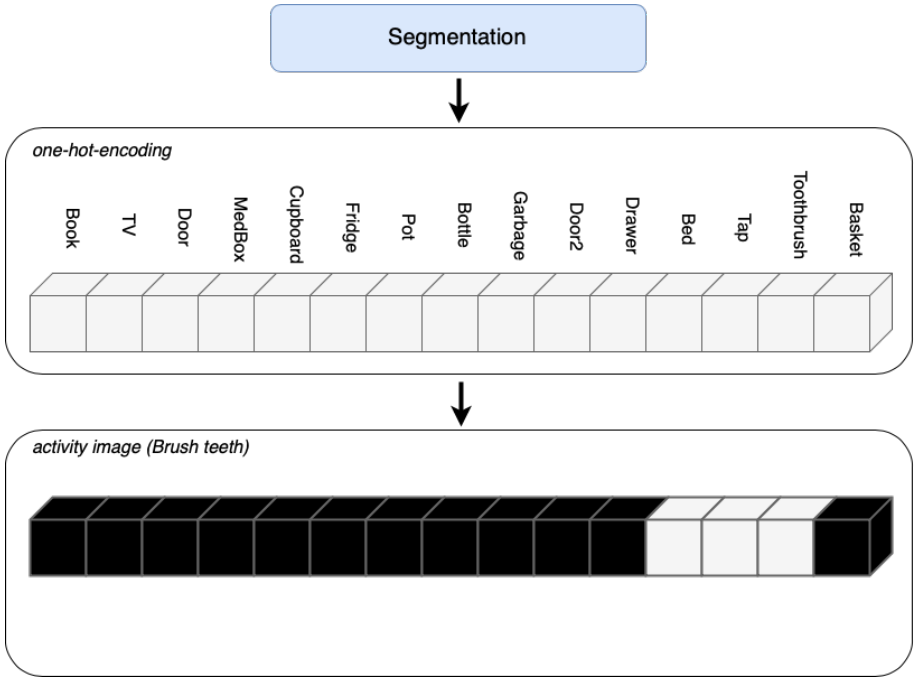

Figure 4.8: Encoding of "brush teeth activity" into vector form.

- As regards the floor pressure sensors, we segment using accelerometer's timestamps and then we map the activation as in the proximity stream using the layout at Figure 4.3 to map the activation of the plates, obtaining $n\_samples$ of dimension $[5 \times 10]$. The process is illustrated in Figure 4.9.

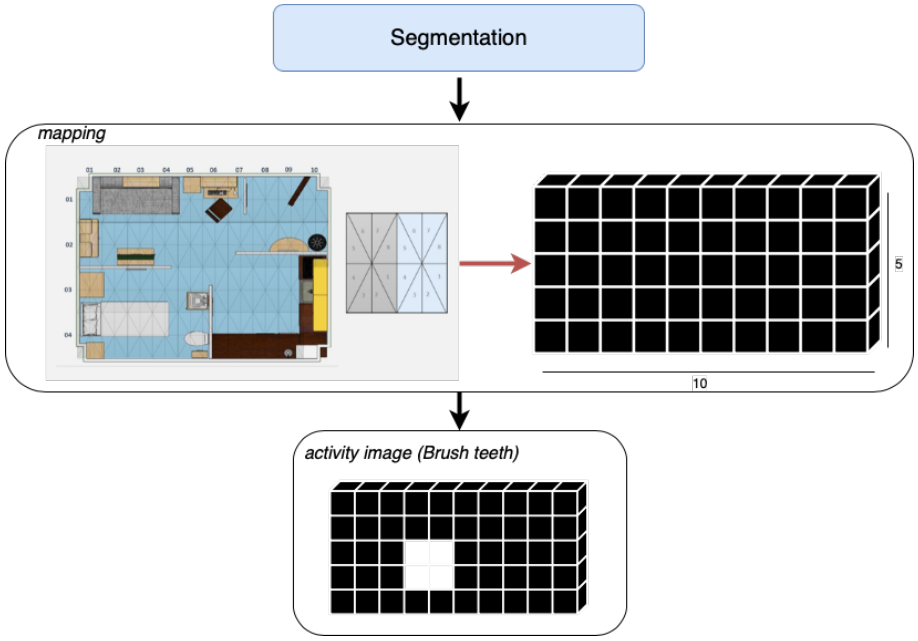

Figure 4.9: Mapping from pressure sensors of a chunk of "brush teeth activity" into matrix form.

4.1.2 Model architecture

As shown in Figure 3.3, we must define the architecture for each source. In this section, we explain the design chosen for each data source.

Accelerometer Architecture

CNNs work well for identifying simple patterns within the data which will then be used to form more complex patterns within higher layers. A 1D CNN is effective when you expect to derive interesting features from fixed-length segments of the overall data and where the location of the feature within the segment is not of high relevance. This applies well to the analysis of time sequences such as accelerometers data.

Figure 4.10 summarizes the architecture of our proposed CNN. The model

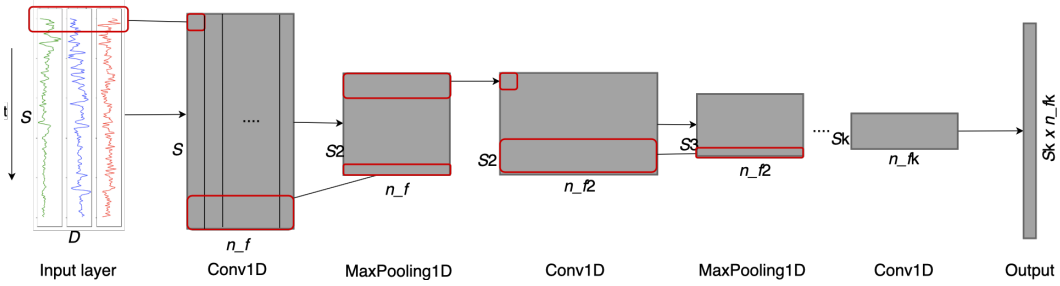

Figure 4.10: Conv-1D architecture for accelerometer data. From the left, the signals coming from the sensor are processed by convolutional and poooling layers. D is the number of channels of the accelerometer, n_f the number of filters.

uses 4 stacked 1D-convolutional layers followed by 1D max-pooling. All convolutional layers use receptive fields of length $(S/5) \times 2$, where S is the size of the sample, and stride of 1 pixel with $ReLU$ activation function. The number of filters n_f for the first three convolutional layers is $18, 36$ and 56, while the last one has $n\_f = 12$ in order to reduce the size of the output to fit well during the fusion with the other sources. Pooling size is again calculated as $(S/5) \times 2$ with a stride of 2 pixels. The output of the last convolution is then flattened and presented as the output of the model.

Binary sensors Architecture

Because we process this stream as a 3D structure, we can treat each sample as a sequence of frames, where each frame defines the full state and position of the binary sensors. Almost like if we were recording the scene from above the smart home. We apply the same architecture commonly used to classify videos. This is carried out making use of a wrapper called *Time-distributed* layer that allows us to apply a layer to every temporal slice of an input, enabling temporal convolutional processing of video tensors.

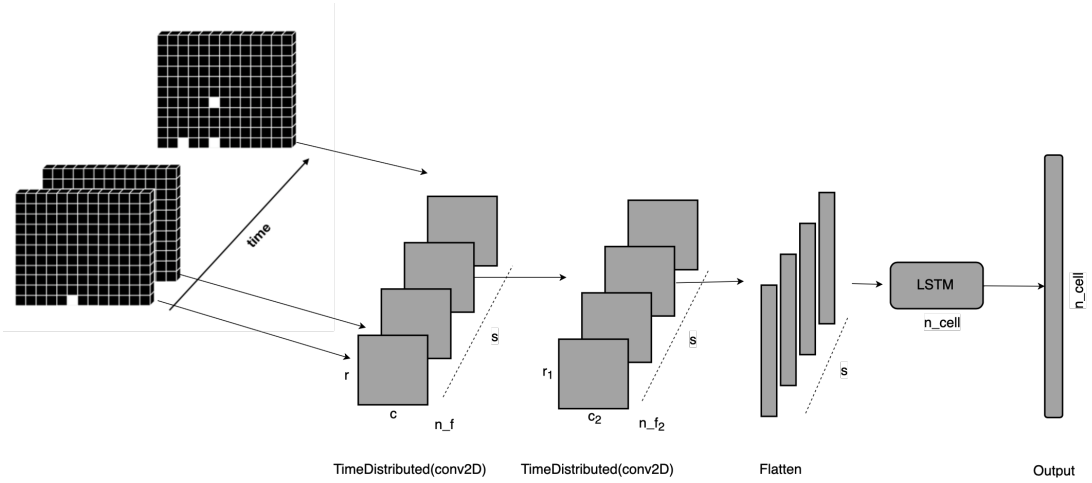

Figure 4.11: Conv-2D architecture for binary sensor data. The input layer is fed to temporal convolutional layers and then passed to an LSTM unit. The last activation is flattened to be returned as output.

Applying two stacked 2D-convolution in this manner, we can extract an arbitrary amount of feature vectors S that hopefully describe the evolution of actions along time. These vectors can then be processed using an LSTM to output an abstract representation of the stream. We decided not to apply pooling as suggested in [40] because the input of the network is constrained by the sliding window mechanism, thus limiting the possibility of downsampling the data, given that Conv-LSTM requires a data sequence to be processed by the recurrent layers. Figure 4.11 illustrates the architecture.

Floor sensors Architecture

The floor samples should represent the route of the user in a range of time inside the smart home. We decided to apply two stacked Conv2D without any pooling to not further reduce the input. The amount of filter are n_f_1=*18* and n_f_2=*16* respectively. The activation map is then flattened to be used afterwards in the fusion step (see Figure 4.12).

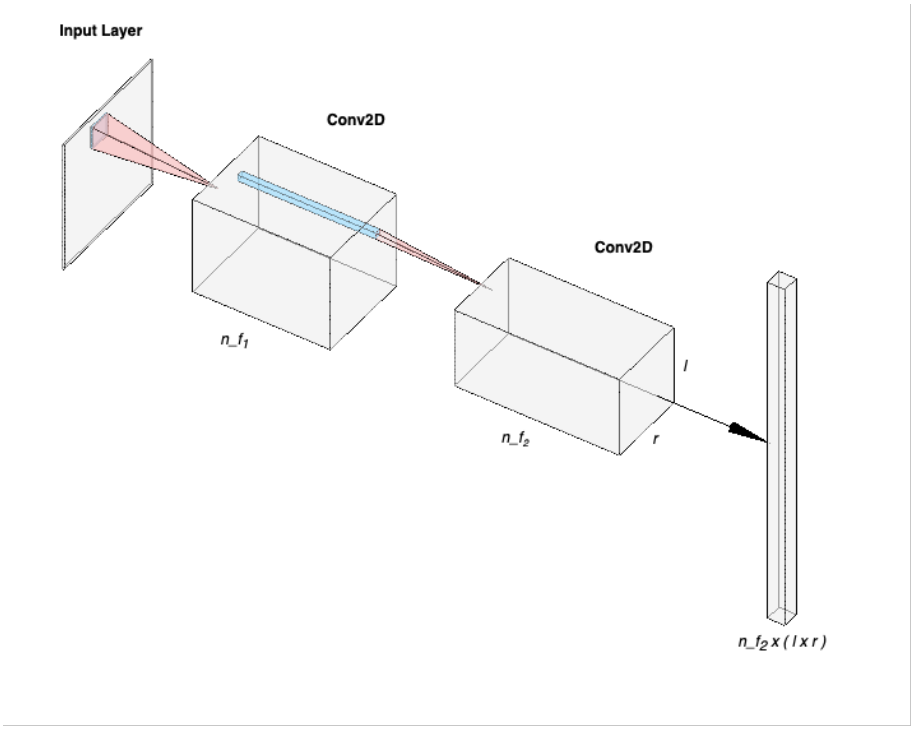

Figure 4.12: Conv-2D architecture for floor pressure sensor data. The input layer is fed to two stacked convolutional layers and then flattened for the output.

To effectively accomplish data fusion, we then merge each output of the models using a horizontal concatenation operation. The complete overview of the model is shown in figure 4.13.

In addition to the inputs previously described, we added a *Time Input* representing the hour of the day in which the sample took place. This feature is represented as a two-value vector calculated as:

$$time\_input = \left[\sin\left(\frac{2\pi \times \text{hour}}{24}\right), \cos\left(\frac{2\pi \times \text{hour}}{24}\right)\right] \tag{4.2}$$

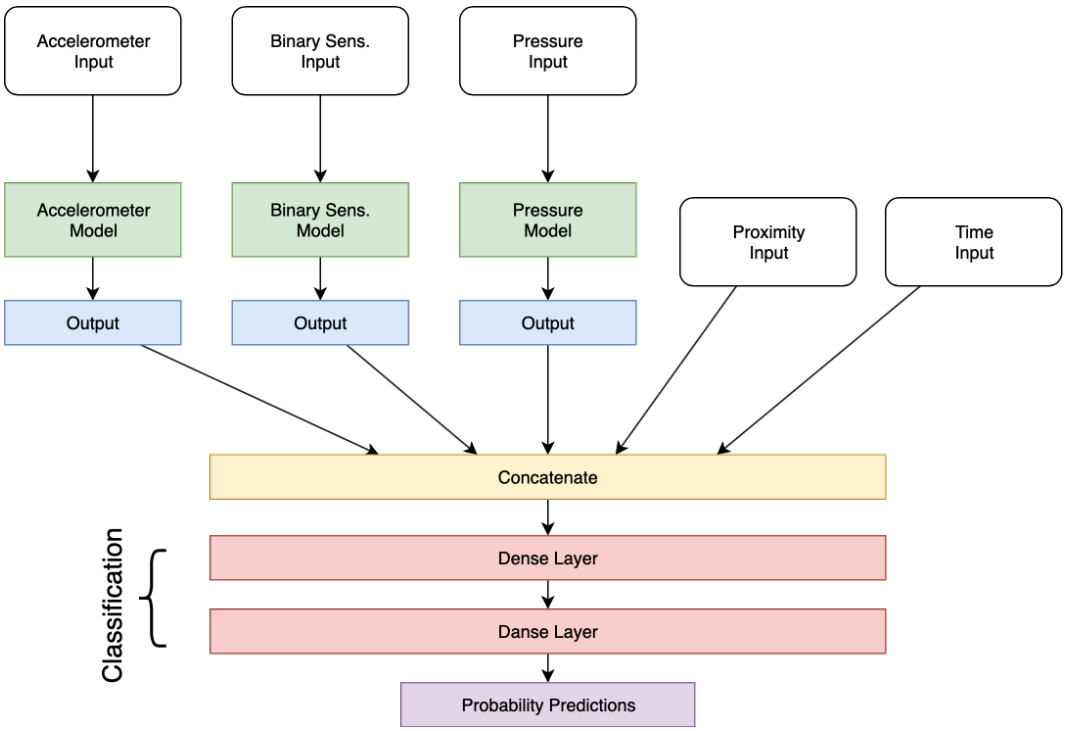

Figure 4.13: Complete model for the multi-input Neural Network. Each different source goes through its individual model producing an Output. Then the fusion step is implemented by merging all the outputs. Finally two dense layers enable the actual classification.

It is a well-known approach for dealing with cyclic features. The idea behind this is simple: we want to induce a prior information about the process in our model, i.e. the end of the cycle is the beginning of a new one. Thus, after applying this mapping, the distance between 00:00 and 23:00 will become smaller.

We note that proximity and time inputs are connected to the concatenation layer by a fully connected layer used to extend their representation to better fit with the other sources during fusion.

4.1.3 Dataset problems

Going through the training data, it is easy to spot sensor data that cannot possibly be accurate. For example, the floor capacitance data shows that the user was "jumping" from the bedroom to the kitchen and back in less than one second. Due to basic physics laws, it is impossible for one person to open the Pajamas drawer while being in the kitchen. Other times the user was supposed to be in the living room, while the binary pressure sensor on the bed or the motion one for the bedroom were triggered. We figured that if this kind of errors were happening both in the Training and Testset, would have been counterproductive to blindly remove them so that the model could try to make predictions using noisy data.

Other problems were related to timestamps, for example, it happens in many days that the streams go back in time or even that time for some of the sources jumps in the future. Many times, windows taken from the proximity sensors were null, meaning the user wasn't near any sensor during some activities; this is obviously wrong because many classes of activity require the user to be using objects or at least to be in the vicinity of them. Synchronization errors may be due to latency in the collection of the data but

are unavoidably destructive for our model. We tried to re-sort or resample the timestamps, but accuracy in prediction dropped so not knowing how to threat this errors, we simply eliminated this portion of the data, reducing even more the amount of data available.

Other issues we faced were due to the difference in the approach of labelling between Train and Test Set, in fact, while in the Training the label was given as a range of time, in the Test set, they ask to predict the activity over blocks of 30 seconds. This brought problems when two different activities happened in the same block. A simple solution would have been to use the same approach as in the training.

Moreover, in the Test set, they require to classify block of times where not every sensor was active; we simply eliminated these inconsistencies using the alignment algorithm. This reduced, of course, the number of samples on which the model is evaluated, but was mandatory because in our model we require every sensor to be working.

4.1.4 Experimental Results

In this section, we evaluate the performance of our model using the UJAmI Dataset. We propose different evaluation metrics to give a complete overview of the strengths and weaknesses of the framework in a real-case scenario. In the next chapter, we will comment on the results, trying to justify their origin.

UCAmI Cup

In Figure 4.14 is presented the histogram of samples in the train set for each class, after alignment and segmentation, using overlapping windows of 50% and T=*5s*. The total amount of samples is 11356.

Method	**Accuracy**	**F-1 Score (weighted)**
Single model *(T=5s)* w/Activity 0	55%	0.5212
Single model *(T=5s)*	63%	0.6543
Average Ensemble model w/Activity 0 *T=5,6,10,15,30*	64%	0.6237
Average Ensemble model *T=5,6,10,15,30*	**73%**	**0.7567**

Table 4.3: Performance measures of the model on Test set. We report results both including or not, the *null activity* 0.

In the Test Set, after the *alignment* (procedure 1) and the removal of one of the measurament due to missing data, we are required to classify 72 blocks of *30s*. If segmented using *T=5s* their distribution is as shown in Figure 4.15.

By using *k-fold validation* [41], an evaluation approach that involves randomly dividing the Train Set into k groups that are used in turn to fit and evaluate the model, we were able to fine-tune the *hyper-parameters* of the network. In figure 4.16 we present the accuracy curve by varying the window size T with the single model:

In Table 4.3 we report the classification accuracy obtained on the Test set using our model. Additionally, we provide F1-score to counter class imbalance by weighting classes according to their sample proportion $\omega_i = n_i/N$ with n_i being the number of samples of the $i-th$ class and N being the total number of samples [36]:

$$F_1 = \sum_i 2 \times \omega_i \frac{precision_i \cdot recall_i}{precision_i + recall_i} \tag{4.3}$$

where precision is defined as $\frac{TP}{TP+FP}$, and recall corresponds to $\frac{TP}{TP+FN}$.

In Figure 4.17 we report the confusion matrix obtained on Test Set with *Average Ensemble model (T=5,6,10,15,30)*. By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group

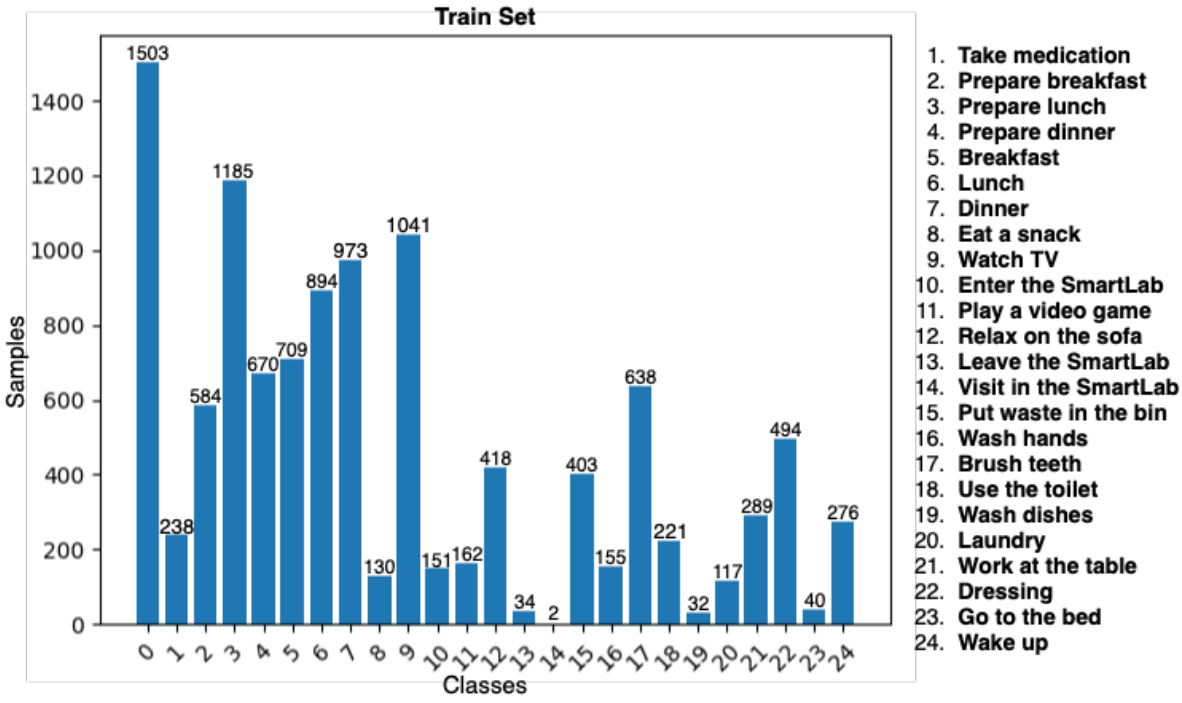

Figure 4.14: Train histogram: segmented using windows of size T=*5s* and overlap=50%

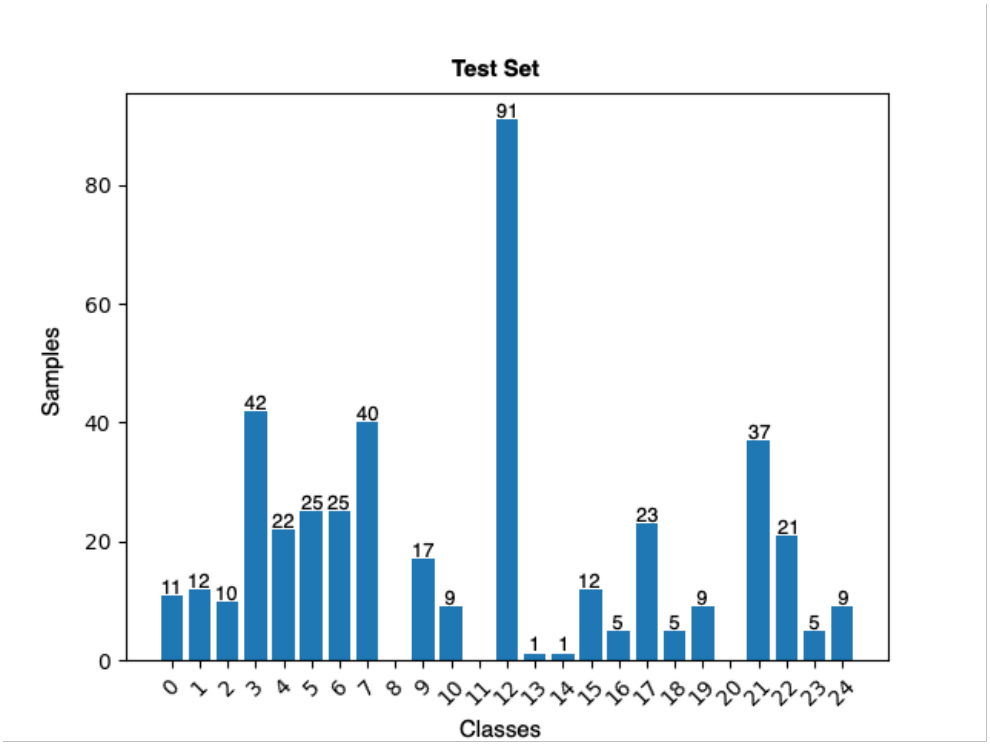

Figure 4.15: Test histogram: segmented using windows od size T=*5s*

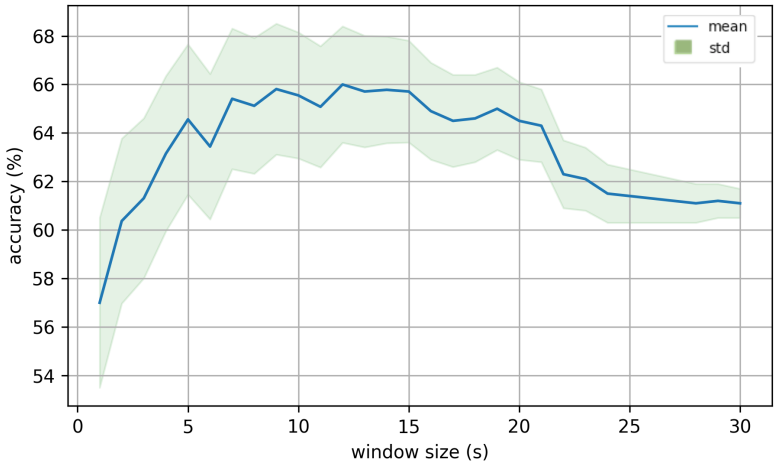

Figure 4.16: Mean accuracy obtained by the model varying window size and using 5-fold cross validation on Train Set. In green the std deviation.

i but predicted to be in group j. Additionally we normalized its elements to better outline the peacks.

In Figure 4.16 is clear that the best performances were obtained by single models using $T>5s$ and $T<22$. Nonetheless we decided $T=(5,6,10,15,30)$ as windows' sizes for our model because they satisfy Eq. 3.6 and we can justify the slightly worse performances of $T=30$ by the significantly smaller amount of samples used to train it. Moreover, the *std deviation* shows a clear stabilization of the results by adding more seconds to the samples. The advantage od a larger window size is the decreased noise with respect of a small sample size.

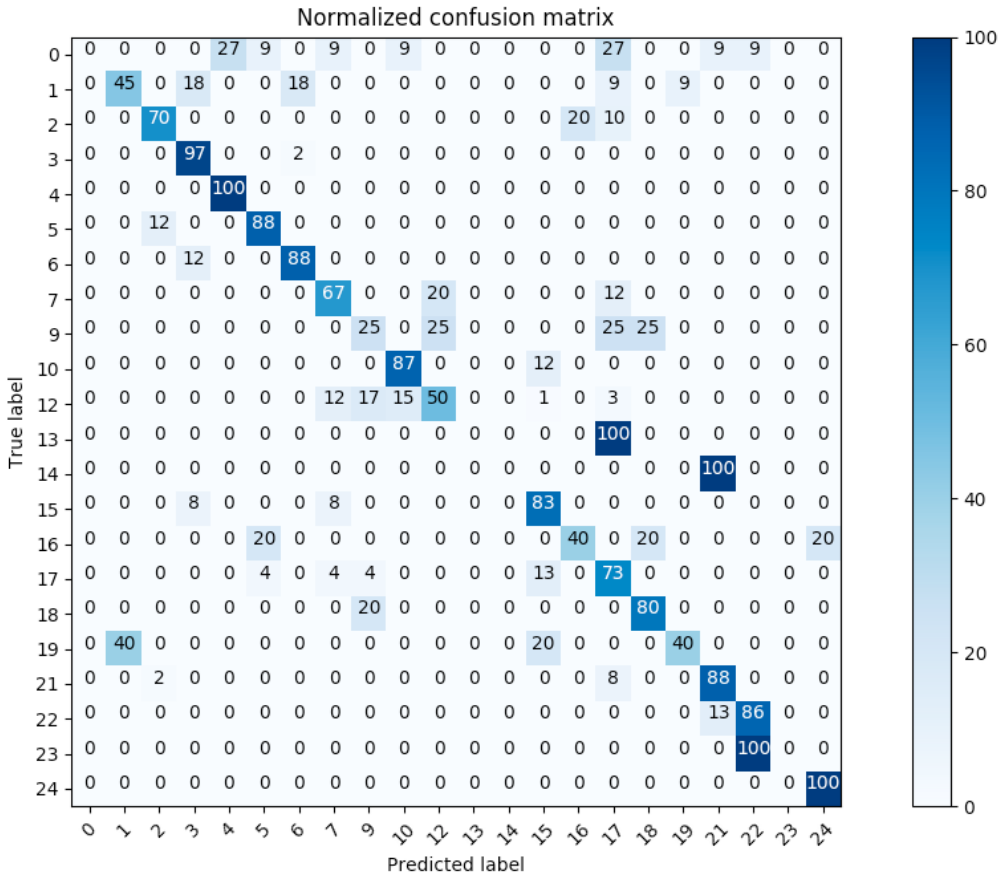

Figure 4.17: Confusion Matrix obtained on Test Set with *Average Ensemble model* (T=5,6,10,15,30). Note the class amount: *13:1, 14:1, 23:4*

Chapter 5

Conclusions and Future Work

In this thesis, we demonstrated the advantages of a multi-input neural network architecture based on convolutional and LSTM layers enabling data-fusion to perform activity recognition. By testing our model on the UCAmI Challenge we were able to compare our technics with those from other researchers.

As shown in Table 5.1 they used Random Forset[42], Naive Bayes Classifier[43], Neural Network[44], Markov Models[45] and Finite Automatas[46]. Our framework outperformed other machine learning techniques from other participants' results in the UCAmI dataset by 13%. While it performed substantially worse with respect to the method using *Finite Automata*, our model is more versatile, truly real-time and can generalize better in different scenarios. Moreover, in our framework, there is no need for feature engineering and we can feed it directly on raw data.

Because the Test set was divided by part of days, that is, each measurement was composed by *morning*, *afternoon* and *evening* data, most of the participants designed three or more models, one for each part of the day. This way they were able to reduce the amount of classes on which the individual

Method	**Accuracy**
Markov Model + NN	45%
Random Forest	47%
NeuralNetwork	60.10%
Naive Bayes Classifier	60.5%
Average Ensemble model ***T=5,6,10,15,30***	73%
Finite Automata	**90,65**%

Table 5.1: Summary table for the results obtained by participants of UCAmI Challenge.

models had to be trained. For example, the class *'Breakfast'*, would be only present in the morning data and could be ignored in the other sets. Our framework can be trained to classify the total amount of activities present in the Dataset, this was feasible by including the *time_ of_ day* feature (4.2), allowing the model to discern similar activities *(e.g. Breakfast, Lunch, Dinner)* by their timestamps.

Our model is *error-resistant*, this can be proven by the fact that training it with data where errors discussed in Section 4.1.3 were removed, the accuracy dropped.

During the development of this work, we discovered how crucial was the synchronization of the sources. This can be seen in Figure 5.1 where we display the sequence of predicted samples over the correct ones for day *2017-11-09 (afternoon)*. Although the predictions are mostly right we can see the majority of errors are at the boundaries of the activities. As a measure of this error factor, we re-evaluated the performance using some *slack*, that is, we counted as correct all those samples where the predicted activity coincided with the sample before or after the one we were classifying. Accuracy went up by 10%.

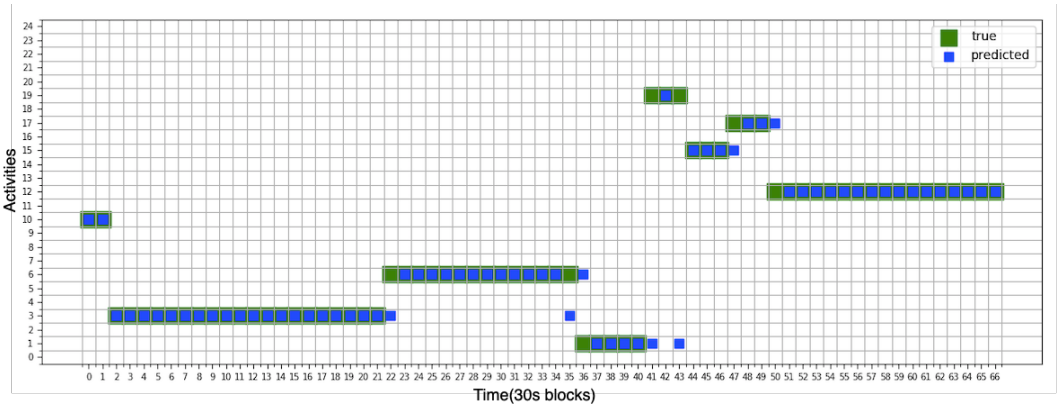

Figure 5.1: Sequence of predicted samples over the correct ones on day *2017-11-09 (afternoon)*.

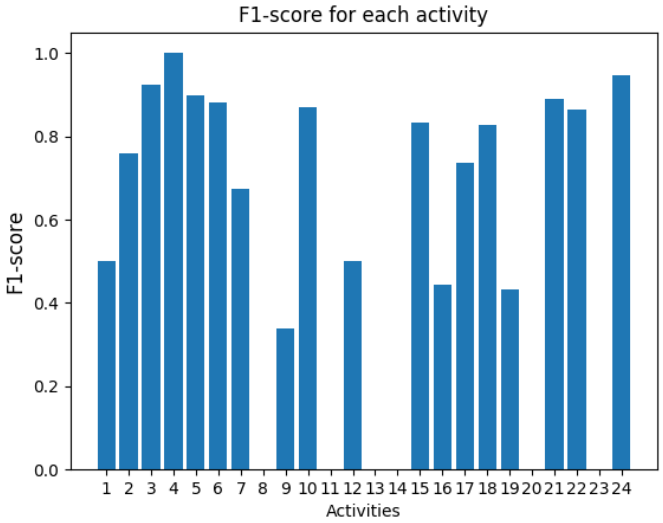

Figure 5.2: F_1-score for each activity in the Test set. Note the sample amount: *8:0, 11:0, 13:1, 14:1, 20:0, 23:4* .

5.0.1 Future Work

One of the crucial problems that we face when tackling any machine learning problem is the problem of unbalanced training data. This is a major obstacle in the UJAmI challenge where the *null* activity is 14% of the whole amount of data and 10 classes out of 25 have under 160 samples, including activity 14, 19 and 13, with respectively 2, 32 and 34 samples (*using T=5s and 50% overlapping*). Another problem for our model or deep learning approaches in general is the discrepancy between the distribution of classes in Train and in Test set. As an example, we can look at activity 12. It falls in the minority set in Training set but is one of the most represented in the Test set, in fact, it was carried out just once during training. Moreover, activity 12(*Relax on Sofa*) is very similar to activities 9(*Watch TV*) and 11(*Play Videogame*) and as a result they are misclassified the most as is shown in the confusion matrix in Figure 4.17. We even tried merging together the samples from this activity and as a result accuracy raised to 80%.

As shown in Table 5.2, the measurement with lowest partial accuracy was *2017-11-21* (*evening*), because all sample from activity 12 were misclassified, in fact, taking a look at the data, in that range of time the binary sensors active were [$SM3$, $SM4$, $SM5$], while the last one is the pressure on the sofa, the other two are respectively the motion sensor in the bedroom and in the bathroom, revealing a faulty detection by the devices.

As another measure of the performances of our model, in Figure 5.2, we present the F1-score calculated for each individual class. We acknowledge that accuracy performances are far from ideal to be used in a scenario for monitoring elderly or patients at risk, but we believe, that with enough data and maybe using some sort of hardware synchronization at the source, we would be able to reach remarkable results that could be used in real-life

Day	Part of day	Accuracy
2017-11-09	morning	90.69%
	afternoon	89.33%
	evening	67%
2017-11-13	morning	74.1%
	afternoon	75%
	evening	65%
2017-11-21	afternoon	66.67%
	evening	41.67%

Table 5.2: Partial Accuracy calculated for each measurement.

scenarios.

In our framework, each model belonging to the ensemble was trained separately and then the predictions were averaged, without giving priority to any of the models. As a future work, we plan to find a way to efficiently train all the individual model together, this way, in the backpropagation step, the best model for each class of activity should be given more credibility in order to achieve better performances.

Bibliography

[1] EUROPEAN ECONOMY. "Economic and Budgetary Projections for the 28 EU Member States". *The 2018 Ageing Report*, 2018.

[2] Steve Lauriks, Annika Reinersmann, Henriëtte Geralde Van der Roest, FJM Meiland, Richard J Davies, Ferial Moelaert, Maurice D Mulvenna, Chris D Nugent, and Rose-Marie Dröes. "Review of ICT-based services for identified unmet needs in people with dementia". *Ageing research reviews* 6, pp. 223–246, 2007.

[3] Xin Hong, Chris Nugent, Maurice Mulvenna, Sally McClean, Bryan Scotney, and Steven Devlin. "Evidential fusion of sensor data for activity recognition in smart homes". *Pervasive and Mobile Computing* 5, pp. 236–252, 2009.

[4] Liming Chen, Jesse Hoey, Chris D Nugent, Diane J Cook, and Zhiwen Yu. "Sensor-based activity recognition". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, pp. 790–808, 2012.

[5] James J Lin, Lena Mamykina, Silvia Lindtner, Gregory Delajoux, and Henry B Strub. "Fish,Äôn,ÄôSteps: Encouraging physical activity with an interactive computer game". In: *International conference on ubiquitous computing*. Springer. 2006. Pp. 261–278.

[6] Sunny Consolvo, David W McDonald, Tammy Toscos, Mike Y Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, et al. "Activity sensing in the wild: a field trial of ubifit garden". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2008. Pp. 1797–1806.

[7] Edward S Sazonov, Oleksandr Makeyev, Stephanie Schuckers, Paulo Lopez-Meyer, Edward L Melanson, and Michael R Neuman. "Automatic detection of swallowing events by acoustical means for applications of monitoring of ingestive behavior". *IEEE Transactions on Biomedical Engineering* 57, pp. 626–633, 2010.

[8] Edward Sazonov, Kristopher Metcalfe, Paulo Lopez-Meyer, and Stephen Tiffany. "RF hand gesture sensor for monitoring of cigarette smoking". In: *Sensing Technology (ICST), 2011 Fifth International Conference on*. IEEE. 2011. Pp. 426–430.

[9] Mitja Luštrek and Boštjan Kaluža. "Fall detection and activity recognition with machine learning". 2008.

[10] Federico Bianchi, Stephen J Redmond, Michael R Narayanan, Sergio Cerutti, and Nigel H Lovell. "Barometric pressure and triaxial accelerometry-based falls event detection". *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18, pp. 619–627, 2010.

[11] Ke-Yu Chen, Mark Harniss, Shwetak Patel, and Kurt Johnson. "Implementing technology-based embedded assessment in the home and community life of individuals aging with disabilities: A participatory research and development study". *Disability and Rehabilitation: Assistive Technology* 9, pp. 112–120, 2014.

[12] Diane J Cook and Narayanan C Krishnan. *Activity learning: discovering, recognizing, and predicting human behavior from sensor data.* John Wiley & Sons, 2015.

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

[14] Uwe Maurer, Asim Smailagic, Daniel P Siewiorek, and Michael Deisher. "Activity recognition and monitoring using multiple sensors on different body positions". In: *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on.* IEEE. 2006. 4–pp.

[15] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y Lee, and Tae-Seong Kim. "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer". *IEEE transactions on information technology in biomedicine* 14, pp. 1166–1172, 2010.

[16] Bhaskar Chakraborty, Andrew D Bagdanov, Jordi Gonzalez, and Xavier Roca. "Human action recognition using an ensemble of body-part detectors". *Expert Systems* 30, pp. 101–114, 2013.

[17] Eunju Kim, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery". *IEEE Pervasive Computing/IEEE Computer Society [and] IEEE Communications Society* 9, p. 48, 2010.

[18] Zhenyu He and Lianwen Jin. "Activity recognition from acceleration data based on discrete consine transform and SVM". In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on.* IEEE. 2009. Pp. 5041–5044.

[19] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. "Activity recognition from accelerometer data". In: *Aaai*. Vol. 5. 2005. 2005. Pp. 1541–1546.

[20] Douglas L Vail, Manuela M Veloso, and John D Lafferty. "Conditional random fields for activity recognition". In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM. 2007. P. 235.

[21] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. "Exploring semi-supervised and active learning for activity recognition". 2008.

[22] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. "Deep learning for sensor-based activity recognition: A survey". *Pattern Recognition Letters*, 2018.

[23] Oresti Banos, Juan-Manuel Galvez, Miguel Damas, Hector Pomares, and Ignacio Rojas. "Window size impact in human activity recognition". *Sensors* 14, pp. 6474–6499, 2014.

[24] Javier Ortiz Laguna, Angel García Olaya, and Daniel Borrajo. "A dynamic sliding window approach for activity recognition". In: *International Conference on User Modeling, Adaptation, and Personalization*. Springer. 2011. Pp. 219–230.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". *nature* 521, p. 436, 2015.

[26] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. "Convolutional neural networks for human activity recognition using mobile sensors". In: *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE. 2014. Pp. 197–205.

[27] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015. Pp. 4489–4497.

[28] Wenchao Jiang and Zhaozheng Yin. "Human activity recognition using wearable sensors by deep convolutional neural networks". In: *Proceedings of the 23rd ACM international conference on Multimedia*. ACM. 2015. Pp. 1307–1310.

[29] Charissa Ann Ronao and Sung-Bae Cho. "Human activity recognition with smartphone sensors using deep learning neural networks". *Expert Systems with Applications* 59, pp. 235–244, 2016.

[30] Andrej Karpathy. "The unreasonable effectiveness of recurrent neural networks". *Andrej Karpathy blog*, 2015.

[31] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. "LSTM: A search space odyssey". *IEEE transactions on neural networks and learning systems* 28, pp. 2222–2232, 2017.

[32] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling". In: *Thirteenth annual conference of the international speech communication association*. 2012.

[33] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches". *arXiv preprint arXiv:1409.1259*, 2014.

[34] Abdulmajid Murad and Jae-Young Pyun. "Deep recurrent neural networks for human activity recognition". *Sensors* 17, p. 2556, 2017.

[35] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. Pp. 2625–2634.

[36] Francisco Javier Ordóñez and Daniel Roggen. "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition". *Sensors* 16, p. 115, 2016.

[37] Shaopeng Liu, Robert X Gao, Dinesh John, John W Staudenmayer, and Patty S Freedson. "Multisensor data fusion for physical activity assessment". *IEEE Transactions on Biomedical Engineering* 59, pp. 687–696, 2012.

[38] Chun Zhu and Weihua Sheng. "Human daily activity recognition in robot-assisted living using multi-sensor fusion". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009. Pp. 2154–2159.

[39] Macarena Espinilla, Javier Medina, and Chris Nugent. "UCAmI Cup. Analyzing the UJA Human Activity Recognition Dataset of Activities of Daily Living". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1267.

[40] Francisco Javier Ordóñez Morales and Daniel Roggen. "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations". In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. ACM. 2016. Pp. 92–99.

[41] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[42] Muhammad Razzaq, Ian Cleland, Chris Nugent, and Sungyoung Lee. "Multimodal Sensor Data Fusion for Activity Recognition Using Filtered Classifier". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1262.

[43] Antonio Jiménez and Fernando Seco. "Multi-Event Naive Bayes Classifier for Activity Recognition in the UCAmI Cup". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1264.

[44] Jesús D Cerón, Diego M López, and Bjoern M Eskofier. "Human Activity Recognition Using Binary Sensors, BLE Beacons, an Intelligent Floor and Acceleration Data: A Machine Learning Approach". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1265.

[45] Paula LAGO and Sozo INOUE. "A Hybrid Model Using Hidden Markov Chain and Logic Model for Daily Living Activity Recognition". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1266.

[46] Sergio Salomón and Cristina Tîrnăucă. "Human Activity Recognition through Weighted Finite Automata". In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 2. 19. 2018. P. 1263.