DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Damien BOUCHABOU

Human Activity Recognition in Smart Homes : tackling data variability using context-dependent deep learning, transfer learning and data synthesis

Thèse présentée et soutenue à Brest, le 30/05/2022
Unité de recherche : LAB-STICC
Thèse N° : 2022IMTA0300

Rapporteurs avant soutenance :

Amel Bouzeghoub	Professeur à Institut TELECOM, TELECOM et Management Sudparis
Francois Portet	Professeur des universités à Université Grenoble Alpes

Composition du Jury :

Président :	Stephane Ploix	Professeur des universités à Grenoble INP
Examinateurs :	Monique Thonnat	Directrice de recherches à INRIA Sophia Antipolis
	Amel Bouzeghoub	Professeur à Institut TELECOM, TELECOM et Management Sudparis
	Francois Portet	Professeur des universités à Université Grenoble Alpes
	Christophe Lohr	Maître de conférence à IMT Atlantique
	Sao Mai Nguyen	Maître de conférence à ENSTA Paris
Dir. de thèse :	Ioannis Kanellos	Professeur à IMT Atlantique

Invité(s) :
Benoit LeDuc Ingénieur Recherche à Delta Dore

ACKNOWLEDGEMENT

J'aimerais tout d'abord remercier mon encadrement en commençant par mon directeur de thèse, Ioannis Kanellos, Professeur à IMT Atlantique Campus de Brest, qui m'a fait partager ses brillantes intuitions et sa sagesse tout au long de cette thèse. Je le remercie pour sa gentillesse, sa disponibilité permanente et pour les nombreux encouragements qu'il m'a prodigués.

Je remercie Monsieur Christophe Lohr, Maître de Conférence à IMT Atlantique Campus de Brest, pour sa disponibilité, sa bienveillance, son écoute, mais surtout ses conseils.

Je tiens à remercier Madame Sao Mai Nguyen, Maître de Conférence à ENSTA Paris, pour m'avoir appris à être analytique, critique et rigoureux, tout au long de ce travail de recherche. C'est à ses côtés que j'ai compris ce que rigueur et précision voulaient dire.

Je remercie Monsieur Benoit Leduc, ainsi que la société Delta Dore de m'avoir fait confiance et soutenue dans le cadre de cette collaboration avec IMT Atlantique, dont la thèse est le fruit. Je le remercie pour le suivi et le lien plus concret avec les problématiques terrain qu'il m'a permis d'avoir.

Je souhaite remercier André Thépaut, aujourd'hui Professeur Emérite, pour m'avoir recruté dans le cadre de cette thèse, fait confiance et pour m'avoir partagé son expérience ainsi que sa sagesse. Je le remercie pour l'oreille qu'il a eu lors de nos discussions au sujet de réseaux de neurones lors de nos sorties de courses à pied, mais aussi pour m'avoir fait découvrir et adorer son beau département qu'est le Finistère.

Merci à Jérôme Kerdreux pour m'avoir aidé à mettre en place mes expérimentations au sein du Laboratoire EXPERIMENT'AAL. Je le remercie d'avoir mis à ma disposition ses compétences techniques et son expertise. Je le remercie pour toutes les discussions techniques que nous avons pu avoir durant ces trois années et qui m'ont beaucoup appris, mais surtout pour sa bonne humeur.

Je remercie tout le département informatique Permanent comme Doctorant et Post-Doctorant qui, avec les questions récurrentes, « quand est-ce que tu la soutiens cette thèse ? » et « as tu rendu ton manuscrit ? », m'ont permis de ne jamais dévier de mon objectif final.

J'adresse un remerciement particulier à mes colocataires, Jonathan Chrun, Jean-Jacques Estembre et Benjamin Somers pour tous les merveilleux moments partagés ensemble qui ont rendu cette expérience de thèse encore plus agréable. Je tiens à les remercier plus particulièremnt pour les discussions nocturnes et tardives qui m'ont permis d'avancer dans mes travaux de recherche.

Mes remerciements vont aussi à ma famille et mes amis qui m'ont soutenue, encouragée et poussée à réaliser ce projet de thèse. Merci à : Mes parents et mon frère pour leurs soutien, Stéphanie Chabrol pour son soutien quotidien et sa bonne humeur, Jean Louis De Cervale pour m'avoir aidé et conseillé sur mes lettres de motivations, Elodie Ranchoux pour m'avoir poussé à me lancer lors de mon hésitation, Xavier Yerro, Adrien et Andréa Cibrelus pour leurs encouragements et leurs soutiens durant ces trois années.

RÉSUMÉ EN FRANÇAIS

La maison intelligente est, de nos jours, au centre d'attentions importantes en matière de recherche et de développement, notamment en ce qui concerne les nombreuses possibilités d'applications et de services innovants qu'elle peut offrir en termes de sécurité, d'économie d'énergie, d'amélioration du confort et d'aide à la santé des habitants. Ces dernières années ont ainsi vu émerger une multitude de techniques et d'approches qui prennent systématiquement appui sur le paradigme de l'intelligence artificielle afin de doter les maisons d'aujourd'hui de ces applications et ces services. La capacité à comprendre le mode de vie et les activités de ses résidents, au travers des capteurs domotiques, est devenu un domaine stratégique. Il stimule désormais d'intenses recherches. En effet, pour pouvoir proposer à ses occupants des services adaptés à leurs situations de vie et répondre à leurs besoins, comprendre le contexte ambiant d'une maison est essentiel.

Cette thèse se propose de contribuer au domaine scientifique et technique actuel de la reconnaissance de l'activité humaine dans une maison intelligente au travers de capteurs domotiques. Elle tente, plus spécifiquement, à apporter une réponse à la question suivante :

Comment reconnaître précisément les activités quotidiennes des habitants dans une maison au moyen de capteurs domotiques non-intrusifs ?

Dans le premier chapitre nous proposons une courte introduction à cette thématique et aux défis qu'elle procure. Nous y définissons et montrons notamment la complexité que peut avoir l'activité humaine.

Dans un second chapitre nous proposons un état de l'art plus ciblé sur les systèmes de reconnaissance d'activité humaine pour en relever les enjeux tant scientifiques que techniques et ainsi donner une claire assise à nos propositions. Nous nous y attachons, en particulier, à présenter comment le flux de capteurs est segmenté, représenté et codé. Nous discutons aussi des méthodes de reconnaissance des formes et d'interprétation des données temporelles pour reconnaître des activités à partir de données de capteurs, ainsi que des jeux de données pour la reconnaissance d'activités. Enfin, nous cherchons à identifier certains problèmes ouverts et des questions liées à ce sujet.

Dans un troisième chapitre, nous décrivons l'approche préconisée, le contexte et la méthodologie expérimentale utilisée dans les travaux menés. Nous présentons le codage du flux de capteurs utilisé et les raisons de son choix et nous détaillons les modèles de base qui servent pour nos comparaisons, la préparation des jeux de données, les métriques de comparaison et, enfin, la méthode d'évaluation.

Dans le quatrième chapitre, nous explorons les questions de reconnaissance des formes en ce qui concerne la reconnaissance des activités. Nous évaluons une première approche de la représentation et de l'extraction de caractéristiques qui utilise des méthodes issues des domaines du Traitement Automatique du Langage Naturel et de la Classification de Séries Temporelles. Nous expliquons la nécessité de modéliser la reconnaissance d'activité dans une maison comme un problème plus complexe qu'un simple problème de reconnaissance de formes et la valeur ajoutée espérée par ces idées.

Dans le cinquième chapitre, nous développons des méthodes pour améliorer la représentation des données de capteurs par des techniques de projection dans un espace latent avancé. Le contexte dans lequel un capteur est activé fournit des informations précieuses sur l'activité en cours. Tout comme un mot peut avoir un sens différent selon les autres mots de son contexte proche (dans une phrase, par exemple), l'activation d'un capteur peut avoir un sens différent selon les autres capteurs de la même trace. L'inspiration de notre approche par des méthodes avancées du Traitement Automatique du Langage Naturel vise, justement, à extraire une représentation contextualisée de ces capteurs afin d'aider les modèles à comprendre dans quel contexte d'activité l'activation d'un capteur se produit. Dans cet ordre d'idées, la méthode implémentée adresse une technique de modélisation du langage capable de contextualiser les mots dans une phrase.

Dans le sixième chapitre, nous présentons une série d'études utilisant soit, le contexte temporel, ou le contexte provenant d'activités précédentes, pour améliorer les modèles de reconnaissance d'activité. Les activités humaines sont souvent liées à l'heure de la journée et aux activités précédentes et s'organisent en patterns plus globaux. Nous proposons d'ajouter une représentation de ces connaissances pour aider les modèles à mieux discriminer les activités qui sont similaires en termes d'activation de capteurs ou qui se produisent à des moments particuliers de la journée.

Dans le septième chapitre, nous proposons une méthode pour appliquer des modèles de reconnaissance d'activité dans des maisons réelles, en utilisant des données synthétiques provenant d'une maison numérique jumelle. Les modèles utilisés dans la reconnaissance d'activité exigent des données étiquetées pour apprendre les paramètres. Clairement, un modèle préparé pour une maison ne peut pas automatiquement être utilisé dans une autre maison en raison des différences dans la topologie de la maison et du comportement des habitants. Nous présentons, ainsi, une méthode d'apprentissage par transfert qui nous permet d'utiliser des données étiquetées et générées synthétiquement à partir du jumeau numérique d'une maison réelle, afin de faire apprendre les paramètres du modèle pour celle-ci.

Enfin, dans le dernier chapitre, nous résumons les résultats de cette thèse, résultats qui apportent une notable amélioration par rapport aux travaux actuels. Nous y discutons des limitations des propositions réalisées, ainsi que des orientations futures pouvant s'appuyer sur

ces travaux.

Table of Contents

List of acronyms

AAL Ambient Assisted Living.

ADL Activity of Daily Lining.

AI Artificial Intelligence.

BN Batch Normalization.

BPE Byte Pair Encoding.

CAM Class Activation Map.

CBOW Continuous Bag Of Words.

CLA Cortical Learning Algorithm.

CNN Convolution Neural Network.

CRF Conditional Random Fields.

DBN Dynamic Bayesian Networks.

DDA Data-Driven Approaches.

DT Decision Tree.

DW Dynamic Windows.

EW Explicit Windowing.

FCN Fully Convolutional Network.

FN False Negative.

FP False Positive.

FTW Fuzzy Time Windows.

GAP Global Average Pooling.

GDP Gross Domestic Product.

Grad-CAM Gradient-weighted Class Activation Map.

GRU Gated Recurrent Unit.

GUI Graphical User Interface.

HAR Human Activity Recognition.

HHMM Hierarchical Hidden Markov Model.

HMM Hidden Markov Model.

IoT Internet of Things.

KDA Knowledge-Driven Approaches.

LSTM Long Short Term Memory.

ML Machine Learning.

MLM Masked Language Model.

MLP Multi-layer Perceptron.

NB Naive Bayes.

NLP Natural Language Processing.

NSP Next Sentence Prediction.

PDA Personal Digital Assistant.

PIR Passive Infrared.

ReLU Rectified Linear Unit.

ResNet Residual Networks.

RNN Recurrent Neural Networks.

SDAE Stacked Denoising Autoencoder.

Seq2Seq Sequence-to-Sequence.

SEW Sensor Event Windows.

SVM Support Vector Machines.

TN True Negative.

TP True Positive.

TSC Time Series Classification.

TW Time Windows.

UMAP Uniform Manifold Approximation and Projection.

List of Figures

List of Tables

INTRODUCTION

1.1 Background

Due to cost reduction and recent advances in sensors, Internet of Things (IoT), and communication technologies have transformed our environment into an ecosystem of interconnected objects that can communicate, collaborate, and capture data. These technologies have stimulated the development of intelligent environments. They are making our transportation systems, cars, factories, hospitals, offices, homes, cities, and personal devices smarter, more efficient, more eco-friendly, and more comfortable.

A great deal of research into these intelligent environments is underway for their potential to provide a wide range of innovative applications and services. Smart homes are part of these intelligent environments and have been increasingly studied and developed these last few years for the many possible applications that they can offer in terms of security, energy saving, comfort improvement, and health support.

1.2 The Smart Home Concept

The house of tomorrow is already designed as a "smart house" but smart home technologies have a much longer history than one might imagine. The idea of smarter homes in terms of comfort and convenience dates back to at least the 1890s and early 1900s, when wealthy individuals took advantage of the advent of electricity to create more automated, luxurious, relaxing, and comfortable homes [15]. Nowadays, a smart home is a house containing home automation devices, such as sensors and actuators, that monitor and control its features. These devices create a technological ecosystem interconnected by means of specific communication protocols [24] on which programmed algorithms or scenarios can rely.

Originally, smart home technology was used to control environmental systems such as lighting and heating, but recently, the technology has expanded so that almost every electrical component in the home can be included in the system. Appliances such as dishwashers, washing machines, refrigerators, etc. are becoming connected and equipped with a variety of sensors making them more intelligent. They can thus send back information or be controlled by the system. Moreover, today our various entertainment objects such as televisions, audio systems, etc. and external house objects such as cars, can also be integrated into the system allowing new interactions and possibilities. Today every object of our daily life could be integrated into this ecosystem.

Smart home technology not only turns on and off lights, appliances, or objects on demand. Today thanks to Artificial Intelligence (AI), it can monitor the internal environment and any activities undertaken when the house is occupied. Moreover it can make decisions in regards to observations taken by the system. As a result, a smart home could be able to "perceive" and "understand" its environment as well as the activities and changes that occur within it, thanks to sensors of various kinds (temperature sensor, motion sensor, magnetic sensor, pressure sensor, etc), installed in strategic places (motion sensor above a bed, magnetic sensors on medicine cabinet door, etc) or in everyday appliances and objects.

Once synonymous with complicated installations and inconsiderate expenses, home automation has changed to be a far simpler and easier addition to the modern home. It is increasingly democratized today by technological advancements such as voice assistants (Alexa, Google, Siri, ...), and new solutions have put comfort, security, and energy efficient advancements within the reach of more people than ever before. The home of tomorrow should not only be a place of residence, but a true platform of services and experiences for the user.

1.2.1 A Smart Home For Saving Energy

One of the great challenges facing the world today is global warming; which, could be partly solved by reducing and better using energy resources. A recent report released in 2020 by Energy Outlook [34], an energy intelligence and consulting firm that collects and publishes data and reports on the state of global energy consumption, shows a 2.9% increase in global energy consumption in 2018. This is considered the fastest rate of increase in decades from its usual primary annual average of 1.5% and is expected to rise rapidly in the coming years. After industries, we see that cities, residential areas, and houses are important points of energy consumption. Today's houses are increasingly designed to be more insulated in order to easily regulate and limit the energy consumption of temperature management systems. Moreover, household appliances are less and less energy consuming. However, despite these efforts. Unfortunately, energy consumption in homes is still high, partly due to poor thermal and behavioral management on the part of residents.

The smart home has become a new option to mitigate this consumption. Thanks Artificial Intelligence (AI) it is not only able to automatically regulate the temperature but also to advise and guide the residents on better consumption. A lot of research is underway to allow the house of tomorrow to be less energy consuming through the use of smart home technologies.

1.2.2 A Smart Home For Better Health

Nowadays, the issue of personal dependence has become one of the great global challenges for public health due to the aging of the population and the increasing cost of medical institutions [12]. However, the loss of dependence does not only affect the elder population, but any person

facing – temporary or permanent – reduction in capacities (physical, functional, cognitive, social, emotional,...) [32]. Most often, it translates into a difficulty in carrying out simple tasks, such as cooking, taking medication, going to the toilet, etc., tasks that can quickly make daily life painful, even costly, both in terms of personal or familial economy and, on a larger scale, in terms of national medical institutions. In France, for example, a new part of the Social Security system was created in 2021 to cover the personal dependence cost. The cost of caring for dependent persons represents 30 billion euros per year, or 1.4% of the Gross Domestic Product (GDP) in France [22].

Providing automated services to enable dependent people to live as independently, comfortably, and healthily as possible in their own homes, while limiting the risks associated with their in domo activities, is a necessity. It has already opened up an unprecedented economic field which is trying to find solutions through intelligent environments [3, 31]. Today, the issue of personal dependence is looking for technical solutions by hopefully drawing from the paradigm of Artificial Intelligence (AI) and smart homes to turn future houses into Ambient Assisted Living (AAL) places.

1.3 Recognition of Human Activities: Concepts, and Approaches

In order to provide adequate services and applications, a smart home must "understand" and "interpret" the daily routines of its residents. It is therefore necessary to develop ad hoc human activity recognition techniques, models, and algorithms to track and analyze the behavior of one or more people to deduce their activities. However, these techniques face many challenges, linked to not only the very complexity of human life, but also of the human environment.

1.3.1 Activity Theory

Before entering into an in-depth discussion of activity monitoring, modeling, and recognition, it is useful to distinguish human behaviors at different levels of granularity. The terms of "Activity" and "Action" are often used to describe human behaviors of different complexity and duration. Commonly and intuitively, the notion of human activity appeals to the idea of granularity and hierarchy. This notion is introduced by "activity theory" [20], [25] which proposes the hypothesis that an activity is a hierarchical construction of operations and actions.

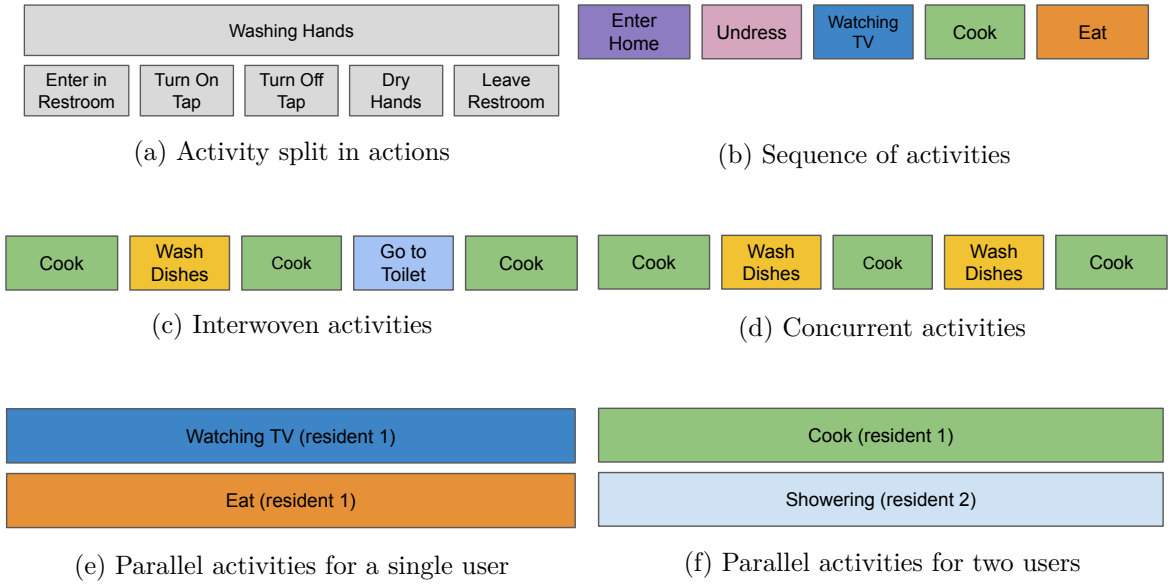
An "operation" refers to a simple, atomic behavior performed by a person over a very short time. For example: turning on or off the light, turning on or off the kitchen tap, turning the door handle, etc.

An "action" refers to an encapsulation of operations. For example: opening a door, which can be composed of multiple operations, turn the handle, push the door, etc.

An "activity" indicates a more complex behavior, composed of actions following and/or

interweaving, or even overlapping. It can be performed by one or more persons and is longer than an action (Figure 1.1a). The more complex the activity is, the greater the number of actions and operations that make it up. Moreover, it is possible that an activity is composed of only one action. For example, we can consider the activity or action of "walking", which makes it difficult to define the boundaries between these two terms.

Humans can performs a very wide range of possible activities that can be grouped in many concepts or families of thought. The concept of Activity of Daily Lining (ADL) was proposed by Katz [18] and then extended by Lawton and Brody [19] to add the concept of instrumented ADLs. This concept refers to human activities that are performed at home daily such as, "bathing", "dressing", "grooming", "work", "eating", etc. This term is used in particular in healthcare to refer to people's daily self-care activities. Healthcare professionals often use a person's ability or inability to perform ADLs as a measurement of their functional status.



(a) Activity split in actions

(b) Sequence of activities

(c) Interwoven activities

(d) Concurrent activities

(e) Parallel activities for a single user

(f) Parallel activities for two users

Figure 1.1 – Representation of activities

The way in which one imagines, or conceives an activity influences the description and / or the division of it. The more generalized the description or vision, the greater the number of actions or operations that comprise it. The activity "watching television" for example, can be viewed in a very general way as "being seated in front of the television" but also, can be divided in more elementary actions like "taking the remote control", "turning on the television", "sitting down", "selecting the program". Even from there it is still possible to go down in level of granularity of the description of the activity by dividing certain actions even more which then become operations. For example, "taking the remote control" which can be cut down to

"seek the remote control with a glance", "walk towards the remote control","pick up the remote control", etc. Describing an activity based on actions, or even operations, can be very complex and depends very much on the level of abstraction or granularity that we choose to describe it.

Once the level of granularity is chosen , it must be taken into account that a person can perform one or more activities in a sequential (Figure 1.1b), intertwined (Figure 1.1c), concurrent, or in a parallel manner. Concurrent and parallel activities seem very similar, but there exists small differences between the two descriptions. Concurrency means executing multiple activity at the same time but not simultaneously, e.g. wash dishes while cooking, see Figure 1.1d. In general, it refers to activities that a resident wants to perform simultaneously but cannot and must alternate between them. Parallel activities means executing multiple activities simultaneously, e.g. watching a TV while eating, see Figure 1.1e.

Moreover, many people do not necessarily live or spend all their time alone. A dwelling may have several people engaged in one or more activities independently, collaboratively, or in parallel (Figure 1.1f). The Figures 1.2 illustrate a hierarchical structure for categorizing human activities according to the number of users and activities involved.

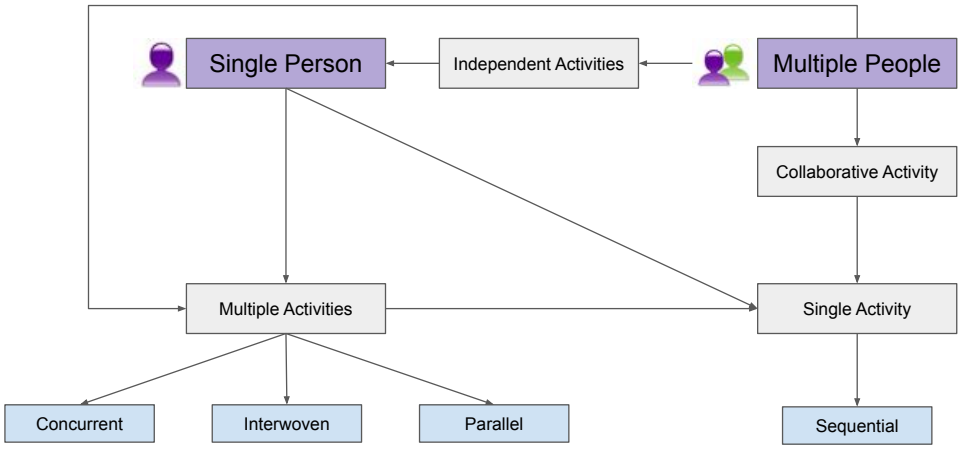


Figure 1.2 – Categorization of human activity

Another factor that must be taken into consideration is the complexity of human activity and that there are a large number of variations for performing the same activity. The same person will not necessarily perform an activity in the same way, nor will it necessarily be performed in the same way by another person. The possibilities become even more numerous when several people collaborate on the same activity. All this shows that human activity is very difficult to describe, interpret, model, and recognize.

1.3.2 Human Activity Recognition Approaches

Monitoring a resident's behavior in a home is a crucial task and involves the use of particular human activity recognition approaches. Generally, human activity recognition systems must capture contextually relevant information to infer an actor's activity, based on the data and data type. Depending on the manner and type of data from these monitoring facilities [17], there are currently two main approaches to recognizing activities [5, 10]: vision-based systems and sensor-based systems (see Figure 1.3).

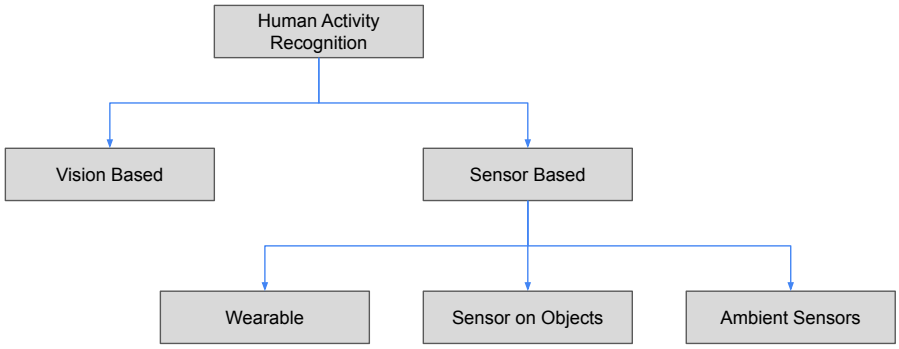


Figure 1.3 – Human Activity Recognition approaches

Vision Based

The vision based human activity recognition uses cameras to track human behaviour and changes in the environment. This approach uses computer vision techniques, e.g. marker extraction, structure model, human body detection, motion segmentation, action extraction, and motion tracking. Researchers use a wide variety of cameras, from simple RGB cameras, to more complex systems with a fusion of several cameras for stereo vision, or depth cameras able to detect the depth of a scene with infrared lights. This domain has been an important research axis for a long time, due to the importance of fields such as human-machine interactions [16], robotics [26], and security [29]. Several survey papers on vision based activity recognition have been published [1, 10]. Beddiar et al. [1] aims to provide an up-to-date analysis of vision based human activity recognition literature and recent progress.

However, these systems pose the question of acceptability. A recent study [30] shows that the acceptability of these systems depends on users' perception of the benefits that such a smart home can provide. It also affects their concerns on the monitoring and sharing of their data. This study shows that older adults (ages 36 to 70) are more open to tracking and sharing data, especially if it is useful to their doctors and caregivers. Meanwhile, younger adults (up to age 35) are rather reluctant to share information. Based on this observation we argue for the use of less intrusive systems, such as smart homes based on IoT sensors.

Sensor Based

human activity recognition from sensors consists of using a network of sensors and connected devices to track a person's activity. The sensors produce data in the form of a time series of state changes or parameter values. A wide range of sensors —magnetic contact detectors, RFID, accelerometers, motion sensors, noise sensors, radar, etc.— can be placed directly on a person, on objects, or in the environment. Thus, the sensor-based solutions can be divided into three respective categories: Wearable [27], Sensor on Objects [21] and Ambient Sensor [14].

Given the privacy concerns of residents and the reluctance of both individuals and legislators to install cameras in what is considered private space, ambient sensor-based human activity recognition systems seem to dominate the smart home research field [8, 17, 2]. Indeed, ambient sensors are generally considered less intrusive and, therefore, are better accepted.

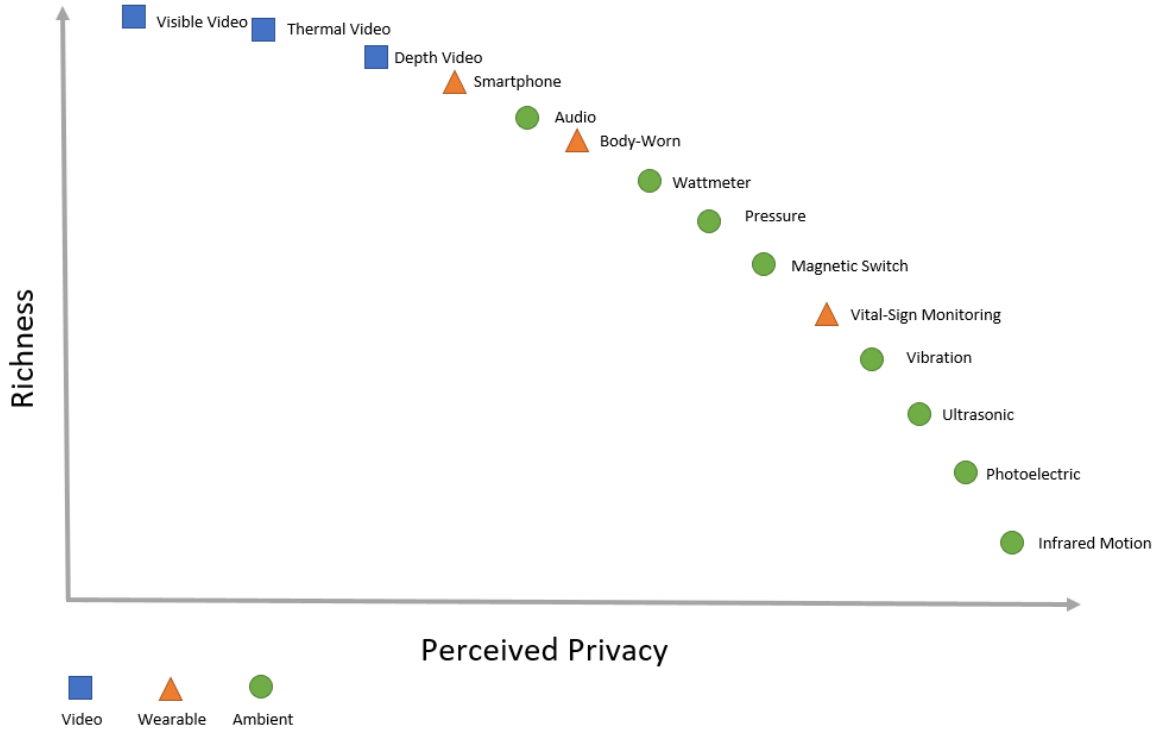


Figure 1.4 – The richness of the sensors versus a user's perceived privacy.

The Figure 1.4 taken from Debes et al. [11] illustrates the richness of the sensors versus the perceived privacy of a person using these sensors. Clearly, sensors that provide rich information about a person's activities are not generally perceived as privacy-preserving. Sensors providing the least amount of information are therefore generally preferred to equip smart homes. This adds additional difficulties and challenges for human activity recognition systems, as we will see in the next section.

1.4 The Challenges Regarding Human Activity Recognition in Smart Homes

In a smart home, the sensors capture into traces the operations, actions, and interactions of the resident(s) within the house, see Figure 1.5. For example, the presence or entrance of a resident into a room can be captured by a motion sensor switching from an "off" state to an "on" state. human activity recognition in smart homes thus consists of translating sensor traces into ADLs. Although simple in its formulation, the recognition of ADLs remains a task with many challenges, both for hardware and the system's algorithms, in terms of design and generalization.

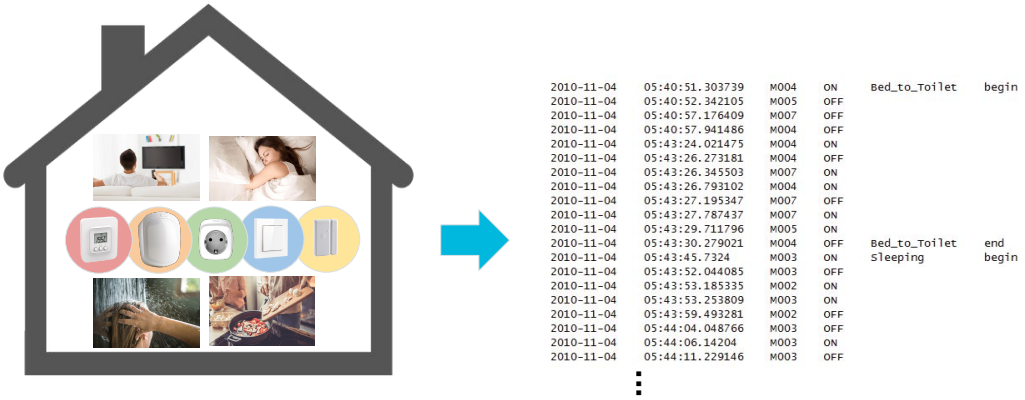


Figure 1.5 – Activities to sensors logs

Human activity recognition systems for smart homes must take into account the different structure, topology, home equipment, resident living habits, and resident number. Because homes and lifestyles are different, the location and number of sensors are also different, which requires adaptability skills.

The number of residents living at the same time in the same house adds other difficulties. The more residents there are, the more the activation traces of the sensors corresponding to different activities are intertwined. For example, if one resident is "cooking" while another is "showering" at the same time, the changes in sensor states that correspond to each activity by one and the other tend to mix. In addition, even if a resident lives alone, other residents on temporary visits may perform activities while present.

Furthermore other challenges arise from the very choice of sensors. As explained in the previous section, ambient sensors, unlike cameras, provide little information. A motion sensor, as its name indicates, will only give the binary information of the existence or nonexistence of a movement in the space it monitors. The activation of each sensor, independently, therefore gives little information about the activity in progress. For example, the activation of the motion sensor in the kitchen may indicate activities such as "cooking", "washing the dishes", "cleaning", etc. Thus, the information offered by this sensor is not very precise, it can only be used in conjunction

with the activation of other sensors to make sense aka by the context of the data. Indeed, it is the context in which a sensor is activated that gives the information meaning.

Moreover, human activity recognition models for smart homes should also look for the context related to the room, the objects, the devices of the house used, and even the type of interactions during ADLs to understand and discriminate them. For example, opening the front door to enter the house and opening the front door to leave the house activates the same type of magnetic sensor, on the same type of object, with the same interaction, but transcribes a different activity.

Furthermore, these sensors are in general triggered by events related to the action of the resident, or an environment change. This generates temporal series that are not regularly sampled (temporally sparse), which makes temporal interpretation difficult and requires, a different treatment from classical temporal series. The activation of two consecutive sensors at an interval of one second or one hour can transmit very different information.

Finally, there is a problem related to labeling sensor activations as activities. Indeed, in order to train a human activity recognition system in a smart home, it is necessary to assign to each sensor activation a label of the activity during which it appears. However, it is often difficult to define a clear cut boundary between the end of an activity and the beginning of a new one. Moreover, the choice of the name of the activity is very dependent on the person in charge of the annotation. For example, the activities "sleeping in a bed at night", or "sleeping on the couch for a nap", can be differentiated by similar labels ("Sleeping" and "Resting") or not ("Sleeping"). In general, a set of activity labels is predefined, such as the ADLs list from Katz [18]'s work for example. Which implies that some activities performed by the resident(s) remain unknown and could be very similar in terms of transferring a sensor's activation traces to labeled activities. It is therefore challenging for human activity recognition systems to find the correct features to discriminate not only all labeled activities between them, but also all unknown ones.

In view of these problems of adaptability, imprecision, robustness, normalization, reliability, etc. of the data, human activity recognition systems in smart homes are continually challenged in terms of pattern recognition and temporal sequence analysis [2]. In an attempt to address all of these challenges and difficulties, researchers are studying different methods for human activity recognition in smart homes.

1.5 Human Activity Recognition Methods for Smart Homes

Algorithms and methods for human activity recognition in smart homes are divided into two broad categories: Knowledge-Driven Approaches (KDA) and Data-Driven Approaches (DDA). KDA uses expert knowledge and rule design. That is, it uses prior knowledge of the domain, its modeling, and logical reasoning. DDA uses user-generated data to model and recognize the activity. It is based on data mining and Machine Learning (ML) techniques.

1.5.1 Knowledge Driven Activity Recognition

In KDA methods, an activity model is built through the incorporation of rich prior knowledge gleaned from the application domain, using knowledge engineering and knowledge management techniques. KDA methods are motivated by real-world observations that involve ADLs and lists of objects required for performing such activities. In real life situations, even if the activity is performed in different ways, the number of objects and types of objects involved do not vary significantly. For example, the activity "brush teeth" may contain actions involving a "toothbrush", "toothpaste", "water tap", "cup", and a "towel". Even if a version of the activity does not include all of the actions such as "cup" or a "towel" it will still include most actions and several key actions such as "toothbrush" and, "toothpaste" will occur every time.

KDA are founded upon the observation that most activities, specifically, routine ADLs, take place in a relative circumstance of time, location, and space. For example, "brushing teeth" is normally undertaken twice a day in a bathroom in the morning and before going to bed and involves the use of "toothpaste" and "toothbrush". Thin implicit relationships between activities, related to temporal and spatial context and the entities involved, provide a diversity of hints and heuristics for inferring activities. The knowledge structure is modeled and represented through forms such as schemas, rules, or networks. KDA modeling and recognition intends to make use of rich domain knowledge and heuristics for activity modeling and pattern recognition. Three sub approaches to KDA exist: mining based approach [28], logic-based approach [9] and ontology based approach [4].

Ontology based approaches are the most commonly used, as ontological activity models do not depend on algorithmic choices. They have been utilized to construct reliable activity models. Chen and Nugent [5] have proposed an overview of this in their work. Yamada et al. [33] use ontologies to represent objects in an activity space. Their work exploits the semantic relationship between objects and activities. A "teapot" is primarily used in the activity of "tea preparation" for example. This approach can automatically detect possible activities related to an object. It can also link an object to several representations or variability of an activity. Chen et al. [6, 4, 7] constructed context and activity ontologies for explicit domain modeling.

KDA provide the advantage to formalize activities and propose semantic and logical approaches. Moreover, these representations try to be as complete as possible to overcome activity diversity. However, some of the key limitations of these approaches are the complete domain knowledge requirement needed to build activity models, and the approaches' weakness when it come to handling uncertainty and adaptability to changes and new settings. The approaches need domain experts to generate knowledge and design rules that can break or bypass previous ones. These limitations are partially solved with the DDA approaches.

1.5.2 Data Driven Activity Recognition

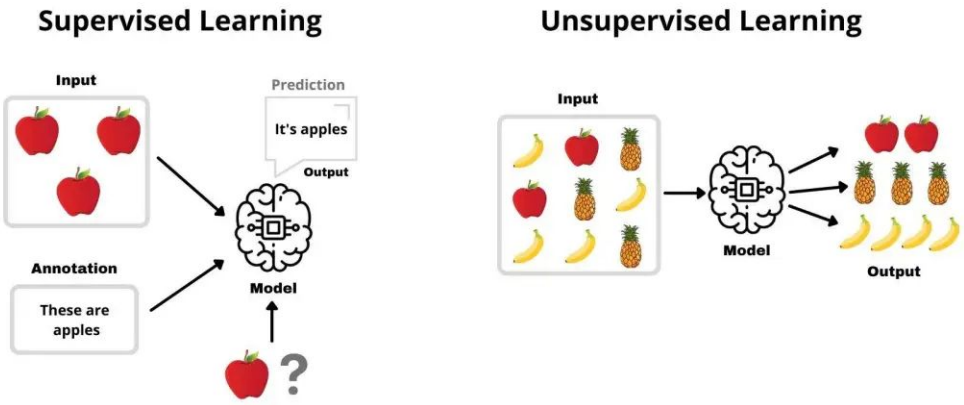


Figure 1.6 – Supervised and Unsupervised approaches

In DDA methods, an activity model is built through the data. DDA are motivated by real-world data that involve ADLs and include both supervised and unsupervised learning methods, see Figure 1.6, based on machine learning and deep learning techniques, which primarily use probabilistic and statistical reasoning.

Supervised learning requires labeled data on which an algorithm is trained. The algorithm uses the labeled data to learn the correlation between the label and the data input. It creates rules that infer the occurrence probability of the label knowing the input. In general, rules correspond to boundaries between data (Support Vector Machines [13]) and discrimination criteria (Decision Trees [23]). After training, the algorithm is then able to classify new unknown data, based on the learned rules. The more variety in examples available, the more the algorithm will be able to generalize and easily handle different inputs.

Unsupervised learning methods extract the knowledge from the data itself without labels. They do not use prior class knowledge and attempt to find and discover logic relations between the data. They need a large amount of data as the number of examples is important. Many unsupervised methods exist, some use distance rules to cluster data that are similar, others attempt to learn these rules by themselves. For strategies to train these types of algorithms the input representation is very important, because it can generate bias. The main advantage of unsupervised learning is that these methods can discover rules and correlations not imagined by an expert.

The DDA's strength is its probabilistic modelling capacity. These models are capable of handling noisy, uncertain, and incomplete sensor data. They can capture domain heuristics, e.g., some activities are more likely to occur than others. They don't require a predefined domain knowledge, are more adaptive to evolution and new situations. However, DDA requires a great deal of data and in the case of supervised learning, clean and correctly labelled data.

1.6 Outline and Research Questions

The progress made in sensors, IoT, communication technologies, and their integration in our daily objects has allowed the development of so-called "intelligent" environments. These technologies open the door to home automation, allowing us to transform and improve our homes into smart homes. Smart homes capable of perceiving (sensors) and acting (actuators) to, for example, save energy and improve our comfort, health and safety. Smart home technology is now more accessible than ever and is becoming more and more popular. However, in order to advance and produce more enhanced services better adapted to each individual, we must overcome a major challenge in development, the ability to "understand" human life, especially ADLs. In this thesis, we are interested in the recognition of human activities in smart homes, and we address the following question:

How to recognize the activities performed by inhabitants in a house through home automation sensors?

Sensors used to equip smart homes must be as non-intrusive as possible to preserve privacy, which implies that they provide poor amounts of information. Each sensor has also been optimised to minimize their energy consumption. Therefore, they output information on their own timing, and are often event-triggered. The data streams of each sensor are unsynchronized between them. Moreover, sensors provide different output value types (binary, scalar, etc). This leads us to our first key research question:

(i) How to efficiently encode, represent, and extract information, features, and patterns from non-intrusive home automation sensors of different natures to perform recognition of the ADLs?

Furthermore, Humans do not always live alone. Different residents performing different activities in parallel mean that a sensor's activation trace that corresponds to an activity could be mixed with another one: this is called the problem of *parallel activities*. This leads us to ask the following question:

(ii) How can we help recognition algorithms to better understand the sequences of sensor activations and sensor relatedness to separate the sequences and better understand captured sensor traces?

Typologies of houses are different, resulting in many different sensor configurations. The DDA is based on data mining, machine learning, and deep learning techniques, which seems promising as this approach allows for better adaptability and generalization. The increase in the number of datasets and thus access to rich data with multiple activity labels, as well as an increase in computing power, allows designers today to generate more complex models. But, even if a deep learning model can optimize generalization on a specific dataset, its performance drops when tested on another dataset, which brings us to the next question:

(iii) Can we transfer the knowledge gained from one house to another house?

Finaly, from an industrial point of view, while models can be trained on any dataset, it is still impossible to obtain a large amount of labeled data from a client. It is not realistic to collect many months of data and ask a customer to label it. This leads to the last question:

(iv) How could we apply activity recognition methods to a actual real house?

1.7 Thesis Organisation

Chapter 2 provides an overview of activity recognition systems in the literature. We present how the sensor stream is segmented, represented, and encoded. We discuss pattern recognition and temporal data interpretation methods for recognizing activities from sensor data as well as datasets for activity recognition. We describe the state of the art developments in the field and discuss open problems and issues related to these topics.

Chapter 3 presents the proposed approach, background, and experimental methodology used in this thesis. We present the sensor stream encoding used in this work. We detail the basic models used for the comparison, the preparation of the datasets, the comparison metrics, and the evaluation method.

Chapter 4 addresses pattern recognition issues with respect to activity recognition. We evaluate the first approach to representation and feature extraction that uses methods from the fields of Natural Language Processing (NLP) and Time Series Classification (TSC). We explain the need to model activity recognition in a house as a more complex problem than just a pattern recognition problem.

Chapter 5 discusses methods for improving the representation of sensor data through advanced integration methods. The context in which a sensor is activated provides valuable information about the current activity. Just as a word can have a different meaning depending on the other words in a sentence, the activation of a sensor can have a different meaning depending on the other sensors in the same trace. We propose the use of advanced Natural Language Processing (NLP) methods to extract a contextualized representation of these sensors to help models understand in which activity context the activation of a sensor occurs. In particular, we present a method inspired by a language modeling technique capable of contextualizing words in a sentence.

Chapter 6 presents studies using previous temporal and activity context to improve activity recognition models. Human activities are often related to the time of day and to previous activities. We propose to add a representation of this knowledge to help models better discriminate between activities that are similar in terms of activation sensors or occur at particular times of day.

Chapter 7 presents a method for applying activity recognition models in real houses using synthetic data from a digital twin house. Models used in activity recognition require labeled

data to learn the model parameters. A model trained for one house cannot automatically be used in another house due to differences in house topology and inhabitant behavior. We present a transfer learning method that allows us to use synthetically generated labeled data from the digital twin of a real house to learn the model parameters for the real house.

Chapter 8 summarizes the results of this thesis and discusses future directions for activity recognition research.

BIBLIOGRAPHY

[1] Djamila Romaissa Beddiar, Brahim Nini, Mohammad Sabokrou, and Abdenour Hadid. Vision-based human activity recognition: a survey. *Multimedia Tools and Applications*, 79 (41):30509–30555, 2020.

[2] Damien Bouchabou, Sao Mai Nguyen, Christophe Lohr, Benoit LeDuc, Ioannis Kanellos, et al. A survey of human activity recognition in smart homes based on iot sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18):6037, 2021.

[3] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes—present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81, 2008.

[4] Liming Chen and Chris Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 2009.

[5] Liming Chen and Chris D Nugent. *Human activity recognition and behaviour analysis*. Springer, 2019.

[6] Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, and Xin Hong. Semantic smart homes: towards knowledge rich assisted living environments. In *Intelligent Patient Management*, pages 279–296. Springer, 2009.

[7] Liming Chen, Chris D Nugent, and Hui Wang. A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, 24 (6):961–974, 2011.

[8] 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(6):790–808, 2012.

[9] Luke Chen, Chris D Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, and Michael Poland. A logical framework for behaviour reasoning and assistance in a smart home. *International Journal of Assistive Robotics and Mechatronics*, 9(4):20–34, 2008.

[10] L Minh Dang, Kyungbok Min, Hanxiang Wang, Md Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108:107561, 2020.

[11] Christian Debes, Andreas Merentitis, Sergey Sukhanov, Maria Niessen, Nikolaos Frangiadakis, and Alexander Bauer. Monitoring activities of daily living in smart homes: Understanding human behavior. *IEEE Signal Processing Magazine*, 33(2):81–94, 2016.

[12] UN Desa. World population prospects 2019: Highlights. *New York, NY: United Nations Department for Economic and Social Affairs*, 2019.

[13] Anthony Fleury, Michel Vacher, and Norbert Noury. Svm-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. *IEEE transactions on information technology in biomedicine*, 14(2):274–283, 2009.

[14] Luis Gomes, Filipe Sousa, and Zita Vale. An intelligent smart plug with shared knowledge capabilities. *Sensors*, 18(11):3961, 2018.

[15] Kirsten Gram-Hanssen and Sarah J Darby. "home is where the smart is"? evaluating smart home research and approaches against the concept of home. *Energy Research & Social Science*, 37:94–101, 2018.

[16] Aashni Haria, Archanasri Subramanian, Nivedhitha Asokkumar, Shristi Poddar, and Jyothi S Nayak. Hand gesture recognition for human computer interaction. *Procedia computer science*, 115:367–374, 2017.

[17] Zawar Hussain, Michael Sheng, and Wei Emma Zhang. Different approaches for human activity recognition: A survey. *arXiv preprint arXiv:1906.05074*, 2019.

[18] Sidney Katz. Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31(12):721–727, 1983.

[19] M Powell Lawton and Elaine M Brody. Assessment of older people: self-maintaining and instrumental activities of daily living. *The gerontologist*, 9(3_Part_1):179–186, 1969.

[20] Aleksei Nikolaevich Leont'ev. Activity, consciousness, and personality. 1978.

[21] Xinyu Li, Yanyi Zhang, Ivan Marsic, Aleksandra Sarcevic, and Randall S Burd. Deep learning for rfid-based activity recognition. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 164–175, 2016.

[22] Dominique Libault. Grand âge, le temps d'agir. *La documentation française*, 2019.

[23] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia Tapia, and Stephen Intille. A long-term evaluation of sensing modalities for activity recognition. In *International conference on Ubiquitous computing*, pages 483–500. Springer, 2007.

[24] Christophe Lohr and Jérôme Kerdreux. Improvements of the xaal home automation system. *Future internet*, 12(6):104, 2020.

[25] Heinrich C Mayr, Fadi Al Machot, Judith Michael, Gert Morak, Suneth Ranasinghe, Vladimir Shekhovtsov, and Claudia Steinberger. Hcm-l: domain-specific modeling for active and assisted living. In *Domain-Specific Conceptual Modeling*, pages 527–552. Springer, 2016.

[26] Roghayeh Mojarad, Ferhat Attal, Abdelghani Chibani, Sandro Rama Fiorini, and Yacine Amirat. Hybrid approach for human activity recognition by ubiquitous robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5660–5665. IEEE, 2018.

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

[28] Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J Patterson. Mining models of human activities from the web. In *Proceedings of the 13th international conference on World Wide Web*, pages 573–582, 2004.

[29] Produte Kumar Roy and Hari Om. Suspicious and violent activity detection of humans using hog features and svm classifier in surveillance videos. In *Advances in Soft Computing and Machine Learning in Image Processing*, pages 277–294. Springer, 2018.

[30] Deepika Singh, Ismini Psychoula, Johannes Kropf, Sten Hanke, and Andreas Holzinger. Users' perceptions and attitudes towards smart home technologies. In *International Conference on Smart Homes and Health Telematics*, pages 203–214. Springer, 2018.

[31] Knud Erik Skouby, Anri Kivimäki, Lotta Haukiputo, Per Lynggaard, and Iwona Maria Windekilde. Smart cities and the ageing population. In *The 32nd Meeting of WWRF*, 2014.

[32] Darcy Ann Umphred, Rolando T Lazaro, et al. *Neurological rehabilitation*. Elsevier Health Sciences, 2012.

[33] Naoharu Yamada, Kenji Sakamoto, Goro Kunito, Yoshinori Isoda, Kenichi Yamazaki, and Satoshi Tanaka. Applying ontology and probabilistic model to human activity recognition from surrounding things. *IPSJ Digital Courier*, 3:506–517, 2007.

[34] Global Energy Statistical Yearbook. Total energy consumption. *Global Energy Statistical Yearbook*, 2019.

Chapter 2

Data Driven-Based Human Activity Recognition in Smart Homes Literature Review

2.1 Introduction

Recognizing human activity in smart homes offers many challenges, as discussed above. Activity recognition systems based on DDA, in particular deep learning based models, are nowadays a mainly studied and promising approach to try to address these different challenges. DDA are indeed able to generalize, show adaptability and do not require a predefined knowledge of the domain.

This chapter provides an overview and discussion of the state of the art of activity recognition systems based on DDA. We will see through the research questions ,recall below, which methods and works exist and try to answer them. We will propose a discussion around previous work limitations and the potential solutions envisaged to remedy them.

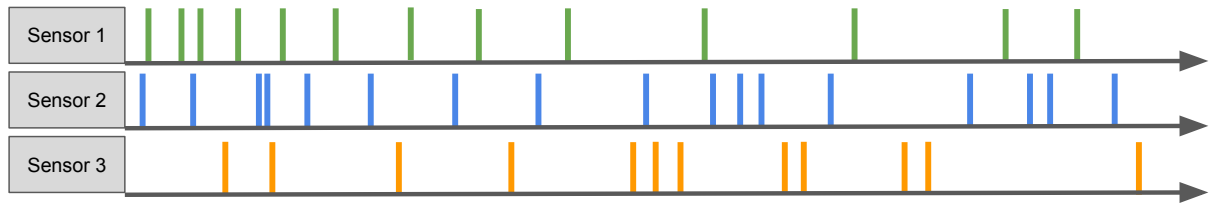
- ***How to efficiently encode, represent, and extract information, features, and patterns from non-intrusive home automation sensors of different natures to perform recognition of the ADLs?***
- ***How can we help recognition algorithms to better understand the sequences of sensor activations and sensor relatedness to separate the sequences and better understand captured sensor traces?***
- ***Can we transfer the knowledge gained from one house to another house?***
- ***How could we apply activity recognition methods to an actual real house?***

Based on these questions, we first present and discuss how the sensor stream is represented, segmented. Then we discuss pattern recognition and temporal data interpretation methods for recognizing activities from smart home sensor data. We discuss about complex human activity

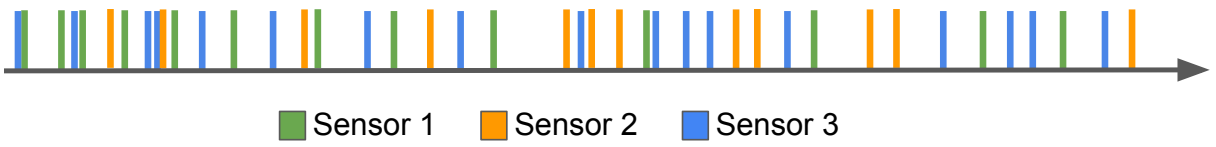
recognition and data variability. We present datasets for activity recognition and discuss their limitations. Finally, we discuss open problems and issues related to this topic

2.2 Representation of Sensor Data Flow

Smart home sensors record the actions and interactions of the residents in their environment over time. These recordings are event logs that capture the actions and activities of daily life in the home. Most sensors are binary or stateful, and only send their status when there is a change, to save battery power and to not overload wireless communications. Indeed, most of the time, to increase ease of installation or to accommodate installation needs in older houses, the sensors run on batteries and communicate over a wireless network. In addition, sensors may have different triggering times. This results in scattered time series samplings and irregular sampling. In general, binary sensors are coded with values of zero and one, while non-binary sensors are normalized and coded with a numeric value between zero and one.



(a) Multivariate representation



(b) Univariate representation

Figure 2.1 – Representation of data flow of binary sensors (each bar is a state change of a sensor)

There are mainly two ways to handle the data provided by home automation sensors, the Figure 2.1 provide an example of the two representation for binary sensors. Either as a set of independent time series corresponding to each sensor, or as a single time series comprised of all the sensor events. In the first case, each sensor records its activity on an independent track over time, see Figure 2.1a. The set of these time series can then be treated as a multivariate time series. This is the most commonly used representation because each sensor is considered as a feature. However, as these time series are made of state changes, they are not continuous (event

according to a resident's action) nor synchronized between themselves (time lag due to wireless communications and sensor reactivity in case of simultaneous activation of sensors).

In general, these time series are resampled in order to generate regular and synchronized time series. Nevertheless, event if resampled time series can gives and encode some information, the sensor duration time activation for example, due to this practice it is possible to alter or modify the data stream. While it is easy to complete the time series by repeating the last state until a new change occurs (see Figure 2.2) the decision over the resampling step may be less obvious. Indeed, the resampling can make certain changes of state disappear if the granularity becomes too large. In contrast, if the sampling is too small, the amount of data becomes very large and more difficult to segment and process. Moreover, this representation makes it difficult to interpret the order of activation relationships between the sensors (order of actions), but allows to easily highlight which sensors are activated at the same time in the house and activation duration time, during an activity.

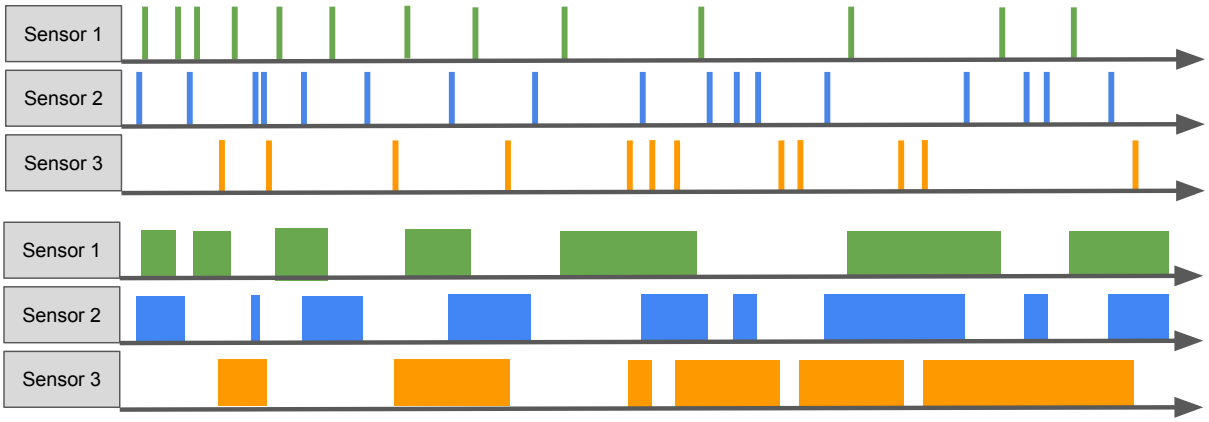


Figure 2.2 – Time series completion of binary sensors example

In the second representation, the set of sensors is perceived as a temporal series of events ordered in time, see Figure 2.1b. This representation has the advantage of not requiring resampling or filling, and thus limiting preprocessing and potential data degradation. It provides the system the ability to use the data in raw form and to exploit the relationships between the sensors and their activation, as well as the order in which they evolve. It also allows the system to compress the information and thus to potentially process longer information sequences. However, it does not allow systems to consider the evolution of time between state changes nor their durations.

2.3 Segmentation of Sensor Data Flow

Once the sensor log is transformed into a univariate or multivariate time series, and as in many areas of activity recognition, a common approach to handling the data is to first segment the data

stream, and then generate features to identify the activity in each of these segments. Depending on the type of activity recognition system, "online" or "offline", some types of segmentation are more suitable than others. A online recognition is a necessity for reactive systems. In some situations, it is not suitable to recognise activities several minutes or hours after they occur, such as in case of an emergency or for danger detection. Online recognition typically uses a sliding window version of the various windowing methods described in the following subsections.

Quigley et al. [58] have studied and compared different segmentation strategies also known as windowing approaches. The Table 2.1 summarizes and categorizes the different segmentation techniques detailed below. An example of three of these segmentation methods applied to a univariate sensors' representation is provided in Figure 2.3.

Table 2.1 – Summary of segmentation methods

Segmentation type	Usable for Real Time	Require resamplig	Time representation	Usable on raw data	Capture long term dependencies	Capture dependence between sensors	# steps
EW	No	No	No	Yes	only inside the sequence	Yes	1
SEW	Yes	No	No	Yes	depends of the size	Yes	1
TW	Yes	Yes	Yes	Yes	depends of the size	No	1
DW	Yes	No	No	Yes	only inside the pre segmented sequence	Yes	2
FTW	Yes	Yes	Yes	Yes	Yes	No	2

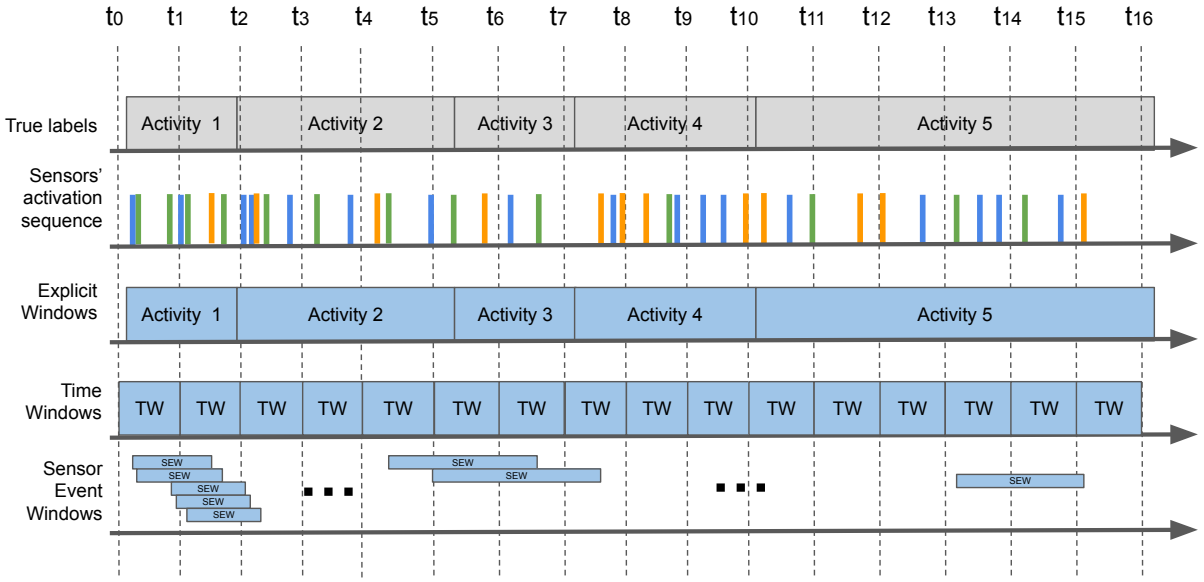


Figure 2.3 – Example of segmentation methods on the univariate sensor representation. In this example, the Sensor Event Windows have a size of 6 events

2.3.1 Explicit Windowing (EW)

Explicit Windowing (EW) consists of parsing the data flow per activity [19, 79]. Each of these segments corresponds to one window that contains a succession of sensor events belonging to the same activity. This window segmentation depends on the knowledge of the data labeling and the size of each window is directly related to the duration of the activity. In case of the absence of labels, it is necessary to find the points of activity change. The algorithms will then classify these windows by assigning the right activity label. This approach has some drawbacks. First, it is necessary to find the segments corresponding to each activity in case of unlabelled data. Then, the algorithm must use the whole segment to predict the activity. It is therefore not possible to use this method for real-time ADLs recognition. Indeed, a real-time activity recognition system tries to determine the activity at the moment it is performed and therefore uses only a fragment of the sequence. However, the EWs contain the entire sensor sequence corresponding to the activity. The system must therefore wait until the entire sensor activation sequence corresponding to the activity has appeared before deciding.

2.3.2 Time Windows (TW)

The use of Time Windows (TW) consists of dividing the data stream into time segments with a regular time interval. This approach is intuitive, but usually only favorable for the time series of sensors with regular sampling over time. This is a common technique with wearable sensors such as accelerometers and gyroscopes. One of the problems with this approach is the selection of the optimal duration for the time interval. If the window is too small, it may not contain any relevant information. If it is too large, then the information may be related to several activities, and the dominant activity in the window will have a greater influence on the choice of the label. van Kasteren et al. [73] determined that a window of 60s is a time step that often grants a good classification score. This value is used as a reference in many recent works [47, 33, 34, 35]. Quigley et al. [58] shows that TW achieve a high level of accuracy but do not often find all possible classes. Indeed, a TW can contain different pieces of activity, especially if the time step is large. The most represented activity within the TW will be used by the system to determine the activity label of the TW. The elements of a very short activity can then be absorbed by the elements of the most present class in the TW. Moreover, in the case of a too small time step, the current TW can be very similar to others which allows the system to make more errors.

2.3.3 Sensor Event Windows (SEW)

Contrary to a TW which divides the sensors stream according to time, a Sensor Event Windows (SEW) divides the stream into segments containing an equal number of sensor events. This type of segmentation is used in the context of a univariate representation of the sensor stream.

Thus, in this segmentation the sensor events are considered as a succession of events, in other words as a sequence of events. Each window is usually labeled with the label of the last event in the window. The sensor events that precede the last event in the window define the context of the last event. This type of window varies in terms of duration. Some Sensor Event Windows (SEW)s can be as short as 5 minutes or as long as 5 hours. The duration of a Sensor Event Windows (SEW) is determined by the delta of time between the first and last event.

This method is simple but has some drawbacks. The size of the window for a number of events, as with any type of window, is a difficult parameter to determine. This parameter defines the size of the context of the last event. If the context is too small, there will be a lack of information to characterize the last event. However, if it is too large, it will be difficult to interpret. A window of 20 to 30 events is usually selected in the literature [5].

Moreover, just like TW, a SEW can contain events that belong to different activities. It is quite possible that the last element of the SEW is an event that belongs to an activity "B" while all previous events belong to an activity "A". In this case, it is possible that the recognition system associates this SEW with activity "A" or instead associates it with activity "B".

2.3.4 Dynamic Windows (DW)

Dynamic Windows (DW) use a non-fixed window size, contrary to the previous methods. It is a two-stage approach that uses an offline phase and an online phase [1]. During the offline phase, the data stream is split into EW. From the EW, the "best-fit sensor group" is extracted based on rules and thresholds define by an expert, in other words, the characteristics, namely, number of activations of each sensor, sensor location, activation duration of each sensor and number of activated sensors. Then, during the online phase, the dataset is streamed to the classification algorithm. When it identifies the "best-fit sensor group" in the stream, the classifier associates the corresponding label with the given input segment.

The DW approach is more reactive in online activity recognition as it may have less instances of sensor events in order to find an activity. For example, the TW approach must run through 60s of sensor data before producing a window, the SEW approach must run through 20 sensor events before producing its first window, but, the DW approach may not always have to wait this long before producing its activity windows. Problems can arise if the source dataset is not properly annotated such that it uses a EW in the first step. Thus, DW inherits the problems of EW. Quigley et al. [58] have shown that this approach is inefficient for modeling similar activities as the "best-fit sensor group" could be also similar. Furthermore, rules and thresholds are manually designed by experts which is time-consuming.

2.3.5 Fuzzy Time Windows (FTW)

Fuzzy Time Windows (FTW) were introduced in the work of Medina-Quero et al. [47]. Unlike previous segmentation methods, FTW not only segments but also extracts features. FTW can be considered as an extension of TW applied to multivariate binary sensor sequences. This segmentation approach uses a succession of FTW of different time sizes defined by the Fibonacci number sequence. For each sensor, the FTW sequences encode, as a numerical value, the activations, more or less long term according to the Fibonacci number sequence. Each FTW evaluates the activation of the sensor in question over a period of time defined by its size and produces a score. These scores are grouped into vectors by sensors, then assembled to form a feature matrix, where each row is a sensor and each column its activation score over time. The column number is related to the number of FTW. The strength of this representation is that it can take into account distant activations of the same sensor in a window.

However, this method has a main drawback, FTW are designed to be only applicable to binary sensors' data. Nevertheless in a smart home the sensors are not necessarily binary, e.g. humidity sensors.

2.3.6 Discussion

The segmentation of the sensor data stream, independently of its representation, is an important step in a human activity recognition system. It depends on the reactivity that one wishes to give to the system. Indeed, EW are more adapted to offline systems while TW, SEW, DW and FTW are more adapted to an online recognition system. We have seen that each segmentation has its advantages and disadvantages. But also that some segmentation methods can only be used with a certain type of representation. Moreover, some segmentation methods like DW or FTW integrate another important step of an activity recognition system, the definition of features. In the next section we will see in more detail this primordial step of a human activity recognition system.

2.4 Feature Extraction

Once the data stream has been segmented, it is necessary to extract features from these segments and give them to a classification algorithm to assign the correct activity label. As seen in the previous chapter, the topology of the houses, their equipment, and the way an inhabitant performs the activities all have a great amount of variability, that increases the feature extraction difficulty. Feature extraction can be done manually following established rules or automatically. It is a decisive step, which must be sufficiently discriminating to help human activity recognition methods to classify the ADLs.

2.4.1 Handcrafted Features

The researchers first defined (see Table 2.2) and studied characteristics [13]. Initially Cook et al. [19] and Yala et al. [79] proposed several feature vector extraction methods, described below.

Table 2.2 – Summary of handcrafted features for smart homes sensor based human activity recognition

Domain	Feature name
Time Features	day of the week hour of day second past midnight time of transition
Windows Features	number of sensor events windows duration (time) most recent sensor last sensor location most frequent sensor dominant location previous dominant location
Sensor Features	count of events elapse time of each sensor relation between sensors sensors localisation

Baseline Method

This consists of extracting a feature vector from each window that contains the time of the first and last sensor events in the window, the duration of the window and a simple count of the different sensor events within the window. The size of the feature vector depends on the number of sensors in the datasets. For instance, if the dataset contains 34 sensors, the vector size will be 34 + 3. From this baseline, researchers upgrade the method to overcome different problems or challenges.

Time Depend Method

This tries to overcome the problems associated with sensor event vanishing influence through time. In most datasets, sensor events are not sampled regularly and the temporal distance of an event from the last event in the segment has to be taken into account. To do this, the sensors are weighted according to their temporal distance. The more distant the sensor is in time, the less important it is.

Sensor Depend Method

This has been proposed to address problems affecting the relationships between the events in a segment. The idea is to weight the sensor events in relation to the last sensor event in the segment. The weights are based on a matrix of mutual information between sensors based on mutual sensor occurrence and calculated offline. If the sensor pairs with the last sensor of the segment in other parts of the sensor flow, then the weight is high and respectably low otherwise.

Sensor Depend Extension Method

This method proposes adding the frequency of the sensor pair to the mutual information matrix. The more frequently a pair of sensors appears together in the dataset, the greater their weight.

Past Contextual Information Method

This is an extension of the previous approaches that takes into account information from past sessions. The classifier does not know the activity of the previous segment. For example, the activity "enter home" can only appear after the activity "leave home". Normally, the previous activity cannot be added to the feature vector because the algorithm might not be able to generalize enough and focus on that particular feature. Therefore, Cook et al. [19] propose a two-part learning process. First, the model is trained without knowing the previous activity. Then, the activity prediction from the previous segment is added to the input feature vector of the classifier when classifying the current segment.

Temporal, Spatial and Frequency Features

Inspired by Cook et al. [19] and Yala et al. [79]'s work, Aminikhanghahi and Cook [5] evaluated different types of sensors flow segmentation and also listed different handcrafted features. Temporal features such as day of the week, time of day, number of seconds since midnight, or time between sensor transitions have been studied. Spatial features were also evaluated such as location and also metrics such as, the number of events in the window or the identifier of the sensor appearing most frequently in the previous segments.

Latent Knowledge Method

This method was proposed recently by Yan et al. [80] to improve the baseline method proposed by Cook et al. [19] and Yala et al. [79]. Their proposal is to learn probability features on sensor occurrence from EWs and combine these probability features with the basic feature vectors that are generated from SEW. To generate the probability features, an HMM was trained to learn the probabilities between sensors in each EW. The authors conclude that the unsupervised learning features significantly improves the performance of hand-created features.

2.4.2 Automatic Features

In the aforementioned works, features are extracted manually. They are carefully designed and heuristic. However, creating handcrafted features is a tedious task that requires expert knowledge. Moreover, handcrafted features lack adaptability.

Unsupervised Machine Learning

A few years ago Cook et al. [19] introduced an unsupervised method for discovering activities from sensor data based on a machine learning algorithm. The algorithm searches for a sequence pattern that best compresses the input dataset. After many iterations, the algorithm reports the best patterns. These patterns are then clustered and given to a classifier to perform the final classification. This approach was a first step for unsupervised feature extraction, but did not produce high performance rates.

Autoencoder

Wang et al. [74] apply a two-layer Stacked Denoising Autoencoder (SDAE) to automatically extract unsupervised meaningfully features. SDAE are an unsupervised artificial neural network from the deep learning domain that learns how to efficiently compress and encode data. Once compressed and encoded it then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. By design, this reduces data dimensions by allowing the system to learn how to ignore the noise in the data.

Wang et al. [74] applyed a two step feature extraction. They first transformed each TW from binary multivariate times series into vectors. They compare two vectors forms: binary representation and numerical representation. The numerical representation method, records the number of firing of a sensor during the TW, while the binary representation method sets the sensor value to one if it fired in the TW. Then these vectors are used by the SDAE which extract a second level of features. This approach has a strong capacity to generate discriminating features shows that automatic features can outperform handcrafted features. However, in this approach there is still a first step of manual generation of characteristics.

Seq2Seq Model

Recently, Ghods and Cook [28] proposed a method also based on a deep learning technique, "Activity2Vec", to generate an activity latent projection from sensor data. They used a Sequence-to-Sequence (Seq2Seq) model [65] to encode and extract automatic features from the sensor's stream. They used EW from an univariate time series representation where each EW is a sequence of sensor ID. The model is trained, to reconstruct each EW in input into output. Their

experiment shows that the "Activity2Vec" method seems generates good automatic features. However, they observed that the intra-class similarity scores were lowest with the handcrafted features, meaning that the handcrafted features appear to perform better than the "Activity2Vec" features.

2.4.3 Discussion

We have just seen different methods and approaches to perform one of the most important steps of an activity recognition system, the feature extraction. Handcrafted feature extraction approaches have proven their worth, but they do not allow for sufficient generalization to address the adaptability issues of the activity recognition system, which is one of the major challenges of this domain. Indeed, once these features are generated by an expert, they are adapted to the environment for which they were made.

Approaches for automatic extraction of these features have also been considered, notably via deep learning methods. Indeed, the automatic extraction of features is one of the challenges taken up by deep learning approaches. However, the envisaged methods do not seem to be sufficiently efficient yet.

2.5 End-to-End Deep Models

In the previous section, we saw that feature extraction is an important step for a human activity recognition system. From these features, an algorithm attempts to identify patterns in order to associate the correct activity label. Therefore, human activity recognition can be reduced to a pattern recognition and classification problem. However, thanks to deep learning, in addition to patterns, these models are able to automatically extract features and perform the final classification task. Deep learning allows us to consider end-to-end models, i.e. models able to extract features from raw data and to perform the classification task at the same time.

CNN models have proven themselves in terms of automatic feature extraction, pattern recognition and classification, especially in the video, image and sound domains.

They have three advantages for human activity recognition: (1) they can capture local dependencies, i.e., the importance of neighboring observations correlated to the current event; (2) they are scale-invariant in terms of step difference and event frequency; and finally (3) they can learn a hierarchical representation of the data.

Therefore, in order to extract features and patterns in activities, Gochoo et al. [30] and Mohmed et al. [50] and Brenon et al. [12] have transformed sensor activity sequences into images and then used 2D CNN.

Gochoo et al. [30] first turned activity sequences into binary images to perform this task. The y axis of the image represents the sensors, which are ordered and grouped by location, and the

x axis corresponds to time. The researchers then designed a Deep 2D CNN to extract features, patterns and finally make the classification. Their work showed that this type of structure could be applied to the ADLs recognition problem. However, this work uses only binary sensors. It could be useful to take into account sensors of other natures. Moreover, other information such as previous activities' labels could also be worthwhile.

In an extension, Tan et al. [68] proposed the use of colored pixels to encode the previous activity and temperature sensors. This work obtained better results than the previous one but neglected other potentially useful information.

Mohmed et al. [50] adopted the same strategy but converted activities into grayscale images. Of note is the fact that previous works did not take into account the activation duration of sensors' activations. They used a gray value to encode the duration of each sensor activation. Then they applied the AlexNet structure [42] to extract patterns and features from images. Finally, they used the features with classifiers to perform the final ADLs recognition.

The previous approach achieved good classification results on EW and a custom dataset. However, the temporal order of sensor activations was not taken into account. Another work of Tan et al. [68] used an image representation with a sliding window on EW to try to tackle this problem. This approach showed good results but appears to not be robust enough to deal with unbalanced datasets (i.e., when some classes have more occurrences than others), or with unlabeled sequences of sensors. Moreover, the temporal dependencies are still not token into account.

Discussion

All these works have shown that CNN are applicable to the ADLs recognition problem, thanks to their ability to extract relevant patterns and features. However, this representation in the form of an image makes it difficult to represent and encode non-binary sensors. Moreover, the temporal aspects and temporal dependencies are not interpretable by a 2D CNN and the order of appearance and evolution of the state changes over time is important information. This is why other methods capable of taking into account this temporal aspect have been and continue to be studied today.

2.6 Sequence Models

The human activity recognition in a smart home, more precisely ADLs recognition is a problem of classification and pattern recognition in time series with irregular sampling. However, more specific machine learning for sequential data analysis have proven efficient to tackle this problem. The Table 2.3 shows a summary and comparison of recent deep learning works.

First, statistical Markov models such as HMMs [17, 16] and their generalisation, probabilistic

Table 2.3 – Summary and comparison of activity recognition methods in smart homes

ref	Segmentation	Data representaion	Encoding	Feature type	Classifier	Dataset	Real-time
Liciotti et al. [46]	EW	Univariate	Integer sequence (one integer for each possible sensors activations)	Automatic	Uni LSTM, Bi LSTM, Cascade LSTM, Ensemble LSTM, Cascade Ensemble LSTM	CASAS [18]: Milan, Cairo, Kyoto2, Kyoto3, Kyoto4	No
Singh et al. [64]	TW	Multivariate	Binary matrix	Automatic	Uni LSTM	Kasteren [72]	Yes
Park et al. [54]	EW	Univariate	Integer sequence (one integer for each sensor Id)	Automatic	Residual LSTM, Residual GRU	MIT [69]	No
Medina-Quero et al. [47]	FTW	Multivariate	Real values matrix(computed values inside each FTW)	Manual	LSTM	Ordonez [52], CASAS A & CASAS B [18]	Yes
Hamad et al. [33]	FTW	Multivariate	Real values matrix(computed values inside each FTW)	Manual	LSTM	Ordonez [52], CASAS A & CASAS B [18]	Yes
Gochoo et al. [30]	EW + SEW	Multivariate	Binary picture	Automatic	2D CNN	CASAS [18]: Aruba	No
Hamad et al. [34]	FTW	Multivariate	Real values matrix(computed values inside each FTW)	Manual	Joint LSTM + 1D CNN	Ordonez [52], Kasteren [72]	Yes
Singh et al. [62]	TW	Multivariate	Binary matrix	Automatic	1D CNN	Kasteren [72]	Yes
Wang et al. [75]	TW	Multivariate / Univariate	Binary matrix, Binary vector, Numerical vector, Probability vector	Automatic / Manual	Autoencoder, 1D CNN, 2D CNN, LSTM, DBN	Ordonez [52]	Yes

graphical models as DBN [57] can model spatiotemporal information. In the deep learning framework, they have been implemented as Recurrent Neural Networks (RNN). Today's RNN shows a stronger capacity to learn features and represent time series or sequential multi-dimensional data.

A RNN is designed to take a series of inputs with no predetermined limit on size. It remembers the past and its decisions are influenced by what it has learned from the past. A RNN can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular neural network, but also by a hidden state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series. But the use of a RNN suffers from the long-term dependency problem [9]. To avoid this problem, two RNN variations have been proposed: the Long Short Term Memory (LSTM) [39] and Gated Recurrent Unit (GRU) [14] that is a simplification of the LSTM.

Liciotti et al. [46] studied different LSTM structures on activity recognition with EW (pre-segmented activity sequences). They showed that the LSTM approach outperforms "Shalow" machine learning-based human activity recognition approaches in terms of classification score without using handcrafted features, as LSTM can automatically generate features that encode the temporal pattern. LSTM approaches and, in particular, bidirectional LSTM, achieve the best results, due to their ability to exploit their internal memories to capture long-term dependencies

in sequences of variable length. They better model the order and density of events, and thus better represent the macro-structure of an activity. A higher performance score granted by LSTM was also reported in Singh et al. [64] work in comparison to traditional "Shalow" machine learning techniques (NB,HMM and CRF).

SEDKY et al. [61] reported that LSTM perform better than AdaBoost, Cortical Learning Algorithm (CLA), HMM or Multi-layer Perceptron (MLP). Nevertheless, the LSTM still have limitations, and their performance is not significantly higher than DT, SVM, nor the stochastic gradient descent of linear SVM when they are associated with good handcrafted features. Indeed, LSTM still have difficulties finding the suitable time scales to balance between long-term temporal dependencies and short term temporal dependencies. A few works have attempted to tackle this issue. Park et al. [54] used a structure with multiple LSTM layers containing residual connections and an attention module. Residual connections reduce the gradient vanishing problem, while the attention module marks important events in the time series. To deal with variable time scales, Medina-Quero et al. [47] have combined the LSTM with a FTW to process the human activity recognition in real time, as FTW can encode short, medium and long-term temporal dependencies. With accuracy ratings lower than 96%, these refinements still need to be consolidated and improved.

Hamad et al. [33] have proposed an extension of the LSTM and FTW structure, by adding FTW using the future data of the sensor in addition to the past information. The purpose of this complement is to introduce a delay in the decision-making of the classifier. The implication is that relying only on the past is not enough to predict the right label for an activity, and that in some cases delaying the recognition time allows the system to make a better decision. To illustrate with an example, if a binary sensor deployed on the front door generates an opening activation, the chosen activity could be "the inhabitant has left the house". However, it may happen that the inhabitant opens the front door only to talk to another person at the entrance of the house and comes back home without leaving. Therefore, the accuracy could be improved by using the activation of the following sensors. It is therefore useful to introduce a time delay in decision-making. The longer the delay, the greater the accuracy, but if this delay is unbounded, the algorithm can no longer be considered real time. While a long delay may be acceptable for some types of activity, others require an extremely short decision time in case of an emergency, i.g the fall of a resident.

The FTW approach allowed systems to obtain excellent classification results. Nevertheless, FTW are designed to be only applicable to binary sensors' data and in a smart home the sensors are not necessarily binary, e.g. humidity sensors.

As the 2D CNN showed capabilities at improving the ADLs recognition problem, the researchers studied a form of CNN more adapted to time series: the 1D CNN. 1D CNN seem to be a competitive solution for time series classification problems as reported in [77, 26]. They allow

good pattern extraction and at the same time interpret the temporal aspect.

The work of [62] shows that 1D CNN can equally be well applied to the ADLs recognition problem and have some advantages against LSTM models. As for all CNN models, 1D CNN have the advantage of being fast to train. They achieve high accuracy, capture patterns, but suffer from the lack of interpretation of long term dependencies, and deal poorly with variable size sequences.

A comparison between a 1D CNN and LSTM approach was performed in the work of Singh et al. [63], Liciotti et al. [46], and Wang et al. [75] . They showed that LSTM perform better in the classification task, as they allow learning long term dependencies from sensor data.

Recently Hamad et al. [34] have combined a LSTM and a 1D CNN with a FTW segmentation. This work has demonstrated that the combination of the two models can help to increase the classification score in general. In addition, this proposed model is capable of achieving better performance for the recognition of minority classes (classes with the least number of occurrences). Nevertheless, the score for the minority classes was still insufficient to handle imbalanced human activity problems.

LSTMs appear to be a viable solution to significantly improve the human activity recognition task in the smart home, despite a longer learning time than approaches based on CNNs. The interpretation of time dependency seems to be the key to human activity recognition systems in smart homes. The combination of an LSTMs based structure with a FTW shows very good performance especially through the idea of using information from the future by embedding a lag in the decision making. However, this approach remains limited to the use of binary sensors

It seems that taking into account the order of occurrence of sensor events (LSTMs), the more or less long term dependencies (LSTMs and FTWs), and the use of more contextual information, especially coming from the future (FTW + lag), increases the performance. As well as the dependencies and relations between sensor activation, the past and future context allow a better interpretation of the current sensor activation.

Like human activity recognition, the field of Natural Language Processing (NLP) deals with sequence analysis problems. Similarly, it seems that word sequence analysis can be related to sensor activation sequence analysis. Sensor activations as well as words have more or less long term dependencies, require past and future contextual information and must respect a certain order.

Syntactic analysis and context modeling have been, very early and naturally, at the center of research in NLP. The latest advances in this field have proposed various unsupervised pre-training methods of projecting words into a vector space, called embedding, to create word representations and language models. These embeddings capture information about the construction of words, sentences and texts. They are also able to capture the context of a word in a document, semantic or syntactic similarities, relations with other words, etc.

Several training structures and methods have allowed advances in the field of NLP. They can be classified into two categories. Static embedding approaches, such as Word2ve [49], GloVe [55], FastText [10], etc. And contextualized approaches, such as ELMo [56], BERT [23], GPT [59], etc. These models have proven to be effective in the context of text classification, and could be considered for the field of human activity recognition in smart homes.

2.7 Complex Human Activity Recognition

One key issue for acticvity recognition is that most sequential data analysis algorithms can only process simple, primitive activities, and can not as of yet deal with complex activities. A simple activity is an activity that consists of a single or few actions caried out by a single user, such as turning on the light, opening a drawer. A complex activity is an activity that involves a sequence of actions potentially involving different interactions with objects, equipment, or other people, for example, cooking. Furthermore, current deep learning models for recognizing human activity in smart homes do not define relations between different activities.

2.7.1 Sequences of Sub-activities

Indeed, ADLs are not micro actions such as gestures that are carried out the same way by all individuals. ADLs that our smart home want to recognize are sequences of micro actions, which we can call compound actions. These sequences of micro actions generally follow a certain pattern, but there are no strict constraints on their compositions or the order of micro actions. This idea of compositionality was implemented by an ontology hierarchy of context-aware activities: a tree hierarchy of activities links each activity to its sub-activities [40]. Another work proposed a method for teaching this hierarchy: as the HMM approach is not well suited to process long sequences, an extension of HMM called Hierarchical Hidden Markov Model (HHMM) was proposed in [8] to encode multilevel dependencies in terms of time and follow a hierarchical structure in their context. human activity recognition in smart home(there have not been extensions of such hierarchical systems using deep learning) uses hierarchical LSTM using two-layers of LSTM to tackle the varying composition of actions. Covering everything from human activity recognition-based on videos proposing [22] to using tow hidden layers in the LSTM for human activity recognition using wearables [76]. This approach can constitute multiple inspirations for human activity recognition in smart home applications. Other works in video-based human activity recognition proposed automatically teaching systems stochastic grammar describing the hierarchical structure of complex activities from annotations acquired from multiple annotators [70].

The idea of these human activity recognition algorithms is to use the context of a sensor activation, either by introducing multi-timescale representation to take into account longer term

dependencies or by introducing context-sensitive information to channel the attention in the stream of sensor activations.

2.7.2 Interleave and Concurrent Activities

Human activities are often carried out in a complex manner. Activities can be carried out in an interleaved or concurrent manner. An individual may alternately cook and wash dishes, or cook and listen to music simultaneously, but could just as easily cook and wash dishes alternately while listening to music. The possibilities are infinite in terms of activity sequence. However, there are some activities whose appearances would be impossible to witness in the dataset as they would be anomalous, such as and individual cooking while the same individual sleeps in their room.

Researchers are working on this issue. Modeling these types of activities is complex, but it could be modeled as a multi label classification problem. Safyan et al. [60] have explored this problem using ontology. Their approach uses a semantic segmentation of sensors and activities. This allows the model to relate the possibility that certain activities may or may not occur at the same time for the same resident. Li et al. [45] exploit a CNN-LSTM structure to recognize concurrent activity with multimodal sensors.

2.7.3 Multi-user Activities

Monitoring the ADLs performed by a single resident is already a complex task. The complexity increases with several residents. The same activities become more difficult to recognize. On the one hand, a group and a resident may interact to perform common activities. In this case, the activation of the sensors reflects the same activity for each resident in the group. On the other hand, everyone could perform different activities simultaneously. This produces a simultaneous activation of the sensors for different activities. These activations are then merged and mixed in the activity sequences. An activity performed by one resident in this case is noise for the activities of another resident.

Some researchers are interested in this problem. As with the problem of recognizing competing activities. The multi-resident activity recognition problem is a multi-label classification problem [3]. Tran et al. [71] tackled the problem using a multi-label RNN. Natani et al. [51] studied different neural network architectures such as MLP, CNN, LSTM, GRU, or hybrid structures to evaluate which structure is the most efficient. The hybrid structure that combines a CNN 1D and a LSTM is the best performing one.

2.8 Datasets and Real-World Applications

Datasets are necessary to train, test, and validate activity recognition systems. Datasets were first generated in laboratories, but these records didn't allow enough variety or complexity of activities and were not real enough. To overcome these issues, public datasets were created from recordings in real homes with volunteer residents. In parallel to the development of practices for being able to allow systems to compare the same condition or train on the same data, competitions were created, such as Evaluating AAL Systems Through Competitive Benchmarking - AR (EvAAL-AR) [29] or UCAmI Cup [24] to improve dataset training.

2.8.1 Real Smart Home Datasets

A variety of public real home datasets exist [69, 72, 18, 2, 20]. De-La-Hoz-Franco et al. [21] provide an overview of sensor-based datasets used in human activity recognition for smart home. They compiled documentation and analysis of a wide range of datasets, with a list of results and applied algorithms. The Table 2.4 details some datasets from the literature, they are the results generated from the hard work of the smart home community.

Table 2.4 – Example of real datasets of the literature

Ref	Multi resident	Resident type	Duration	Sensor type	# of Sensors	# of Activity	# of Houses	Year
van Kasteren et al. [72]	No	Eldery	12-22 days	Binary	14-21	8	3	2011
Alemdar et al. [2]	Yes	Young	2 months	Binary	20	27	3	2013
Cumin et al. [20]	No	Young	2 weeks	Binary, Scalar	236	20	1	2017
Cook et al. [18]	Yes	Eldery	2-8 months	Binary, Scalar	14-30	10-15	>30	2012
Ordóñez et al. [52]	No	Eldery	14-21 days	Binary	12	11	2	2013
Tapia et al. [69]	No	Eldery	2 weeks	Binary, Scalar	77-84	9-13	2	2004

Nevertheless, datasets are not perfect, and dataset production includes several problems such as: sensors type and placement, variability in terms of user profile or typology of dwelling, and the annotation strategy.

Sensor Type and Positioning Problem

When acquiring data in a house, it is difficult to choose the sensors and their numbers and locations. It is important to select sensors that are as minimally invasive as possible in order to respect the privacy of the volunteers [2]. Because of this no cameras or video recordings are used. The majority of sensor-oriented smart home datasets use so-called low-level sensors. These include infrared motion sensors (PIR), magnetic sensors for items that open and close, pressure sensors placed on sofas or beds, and sensors for temperature, brightness, monitoring of electricity, water consumption, etc.

The location of these sensors is critical for properly capturing activity. Strategic positioning allows a system to accurately capture certain activities, e.g. a water level sensor in the toilet to capture toilet usage or a pressure sensor under a mattress to know if a person is in bed. There is no precise method or strategy for positioning and installing sensors in homes. CASAS [18] researchers have proposed and recommended a number of strategic positions. However, some of these strategic placements can be problematic in terms of long term use. It is possible to imagine that during the life of a house the organization of or use of its rooms may change. For example, if a motion sensor is placed above a bed to capture its use but the bed is moved to a different place in the room, then the sensor will no longer be able to capture this information. In the context of a dataset and its use to validate algorithms, this constraint is not important. But it becomes important in the context of real applications to evaluate the resilience of algorithms, which must continue to function in case of loss of information.

In addition to sensors positioning, it is important to choose enough sensors to cover maximum capture of all possible activities. The number of sensors can be very different from one dataset to another. For example, the MIT dataset [69] uses 77 and 84 sensors for each of its apartments. The Kasteren dataset [72] uses between 14 and 21 sensors. ARAS [2] has apartments with 20 sensors. Orange4Home [20] is based on an apartment equipped with 236 sensors. This difference can be explained by the different types of dwellings, but also by the number and granularity of the activities that we want to recognize. Moreover, some datasets are voluntarily over-equipped. There is still no method or strategy to define the number of sensors installed according to an activity list.

Profile and Typology Problem

It is important to take into account that there are different typologies of houses: apartment vs house, dwellings with gardens, multiple floors vs single floor, dwellings with one or more bathrooms, dwellings with one or more bedrooms, etc. These different types and variability of houses create multiple difficulties; for example, the investment in terms of number of sensors can be more or less important for activity recognition. In addition, the network coverage of the sensors can be problematic. For example, Alemdar et al. [2] faced a problem of data synchronization. One of their houses required two sensor networks to cover the whole house. They had to synchronize the data for their dataset needs.

Also, because each house is different, the data from one house to another is very different. Thus a model learned in one house is hardly applicable in another because of differences in house configuration; sensor equipment; and family compositions and habits. Indeed, the location, the number and the sensors type of smart homes can influence activity recognition systems performances. This issue could be solved by collecting a new dataset for each new household to train the models anew, however this is costly.

Another solution is to adapt the models learned in a household to another. Transfer learning methods have recently been developed to allow pre-trained deep learning models to be used with different data distributions, as reviewed in [78]. Transfer learning using deep learning has been successfully applied to time series classification, as reviewed in [25]. For activity recognition, Cook et al. [15] reviewed the different types of knowledge that could be transferred into traditional machine learning. These methods can be updated with deep learning algorithms and by benefiting from recent advances in transfer learning for deep learning. Furthermore, adaptation to new settings have recently been improved by the development of meta-learning algorithms. Their goal is to train a model on a variety of learning tasks, so it can solve new learning tasks using only a small number of training samples. This field has seen recent breakthroughs, as reviewed in [41], which as of yet have never been applied to human activity recognition. Yet, the peculiar variability of human activity recognition data in smart home can only benefit from such algorithms.

It is therefore necessary that the datasets be able to handle different house configurations in order to train and evaluate the algorithms with a high generality of configurations. Fortunately, there are several datasets with several houses. [69, 72, 72, 2]. CASAS [18] is one such dataset, with about 30 different house configurations. These datasets are often used in the literature [21]. However, the volunteers are mainly elderly people, and coverage of several age groups is vital for developers and researchers. After all, a young resident likely does not display the same behavior as an older one. The Orange4Home dataset [20] covers the activities of a young resident. However, the number of residents is also important. Activity recognition is more complex in the case of multiple residents. This is why several datasets also cover this field of research [18, 2, 72].

Annotation Problem

Dataset annotation is something essential for supervised algorithm training. When creating these datasets, it is necessary to deploy strategies to enable annotation. Such as journals [72], smartphone applications [20], Personal Digital Assistant (PDA) [69], Graphical User Interface (GUI) [2] or voice records to annotate datasets [72].

As these recordings are made directly by volunteers, they were asked to annotate their own activities. For the MIT dataset [69], residents used a PDA to annotate their activities. Every 15 minutes, the PDA beeped to prompt residents to answer a series of questions to annotate their activities, however, several problems were encountered with this method of user self-annotation. Such as some short activities not being entered, errors in label selection, or omissions. A post-annotation based on the study of a posteriori activations was necessary to overcome these problems, thus potentially introducing new errors. In addition, this annotation strategy is cumbersome and stressful because of the frequency of inquiries. It requires great rigor from the volunteer and at the same time interrupts activity execution by pausing it when the

information is being given. These interruptions reduce the fluidity and natural flow of activities.

van Kasteren et al. [72] proposed another way of annotating their data. The annotation was also done by the volunteers themselves, but using voice through a Bluetooth headset and a journal. This strategy allowed the volunteers to be free to move around and not need to create breaks in the activities. This allowed for more fluid and natural sequences of activities. The Diary allowed the volunteers to complete some additional information when wearing a helmet was not possible. However, wearing a helmet all day long was a constraint.

The volunteers of the ARAS dataset [2] used a simple GUI on tablets to annotate their activities. Several devices were placed in homes to minimize interruptions in activities and avoid wearing an object such as a helmet all day long. Volunteers were asked to indicate only the beginning of each activity. It was assumed that residents will perform the same activity until the start of the next activity. This assumption reflects a bias that sees human activity as a continuous stream of known activities.

2.8.2 Synthetic Smart Home Datasets

As seen above, the creation of real datasets is a tedious task and recording campaigns are difficult to manage. It requires hardware, volunteers, and post-processing. The cost to build real smart home and the collection of datasets for such scenarios is expensive and sometimes infeasible for many projects. Measurement campaigns must include a wide variety of activities and actors. They should be done with substantial rigor to obtain plenty of qualitative data. Moreover, difficulties with finding the optimal placement for sensors [36], finding appropriate participants [37, 48] and the lack of flexibility [7, 27] makes dataset collection difficult. Even if fully equipped and ready-to-use environments, such as Amiqual4Home [43], can be used on an ad hoc basis, recruiting volunteers and collecting data is still very time-consuming. For these reasons, researchers created smart home simulation tools. The Table 2.5 present and compare some simulation tools from the literature.

Table 2.5 – Overview of smart home simulation tools.

Simulator	Open-Source	3D	Cross-Platform	Approach	Focus	Multi-Inhabitants	Fast-Forwarding
OpenSHS [4]	Yes	Yes	Yes	Hybrid	Dataset generation	Partially	Yes
SIMACT [11]	Yes	Yes	Yes	Model-based	Visualisation	No	Not applicable
PerSim 3D [44]	No	Yes	Yes	Model-based	Dataset generation	No	Not applicable
Park et al. [53]	No	Yes	Yes	Interactive	Visualisation	No	Yes
Ariani et al. [6]	No	No	No	Interactive	Dataset generation	Yes	No
SESim [38]	Yes	Yes	No	Model-based	Dataset generation	No	Yes

These simulation tools can be categorized into two main approaches, model-based [44] and interactive [66], according to Synnott et al. [67]. The model-based approach uses predefined

models of activities to generate synthetic data. [11], Lee et al. [44] and Ho et al. [38] use predefined models of the agent behavior, describing different possible events and activities with their probability of occurrence, their order and their duration. With this first approach the quality of the data relies heavily on the quality of the modeling. These simulators usually consider big granularities of events and activities, as it is complicated to model fine-grained events and activities. Finally, these approaches are usually entirely scripted. If a user wishes to simulate several days (or more) of data, he needs to script each day independently.

In contrast, the interactive approach relies on having an avatar that can be controlled by a researcher, human participant, or simulated participant. [53] and Ariani et al. [6] assume a human controlling an avatar to perform activities in the simulated environment. These approaches allow more coherent and realistic behavior as there is a real human involved. However, they require near-real time interaction between the controlling human and the simulated environment. Thus, producing large amounts of data is time consuming. The usage of these kinds of simulation tools are more restricted to the generation of data during a short period, focusing on specific activities or situations.

Some recent work focusing on hybrid simulators such as OpenSH [4], can combine advantages from both interactive and model-based approaches. The smart home simulation tools can focus on dataset generation or data visualization. Some simulation tools provide multi-resident options or fast forwarding to accelerate the time during execution.

These tools allow researchers and developers to quickly generate data and visualize it. But capturing activities can be unnatural and far too clean, as in they do not contain the messiness of real human actions which are far from perfect in their execution. Systems need the noise generated from human behavior; without it some of the uncertainty found in real human actions may be missing.

2.8.3 Discussion

Real datasets such as Orange4Home [20] provide a large sensor set. That can help to determine which sensors can be useful for which activities. CASAS [18] include many house and apartment configurations, plus multiple topologies with elderly residents. Which allows systems to evaluate their adaptability to different house topologies. ARAS [2] has younger aged residents and multi-residents' living in a single dwelling. These are all useful for validating the noise resilience and segmentation ability of an activity recognition system.

The strength of datasets from real houses is their realism. They are real playgrounds that allow to evaluate algorithms in situations close to reality. But sensors in these datasets may ironically be placed too strategically, as they are focused on covering certain specific kinds of activities. In some datasets, PIR sensors are placed in a grid or installed at checkpoints to track a resident's trajectory. Strategic placement, a large number of sensors, and/or the choice to use

a particular sensor is great for helping algorithms to infer knowledge, but are not indicative of how real house sensor set-up would happen.

Moreover, the current publicly available datasets from these real-world environments are not sufficient to cover the probability distribution of any possible activities, lifestyle or configuration variability. These datasets are indeed limited in the number of activities and occupants they take into account, as well as the types of sensors and the type of resident. Therefore, algorithms trained on data from one particular environment will be difficult or impossible to reuse in others as they are adapted to a particular configuration of house, inhabitants and life habits.

Collecting new datasets is a very expensive and difficult task. The cost of sensors, the difficulty of accessing multiple types of homes, recruitment, collecting data from actual residents also takes time, and cannot be expedited. Moreover, the data requires labeling, i.e., segmentation of the data stream (start and end time of activities) as well as their class labels, which implies a significant investment on the part of the user and presents a high risk of error, as these annotations are performed manually and usually by the user himself.

Synthetic datasets generation tools allow systems to quickly evaluate different configuration sensors and topologies. In addition, they can produce large amounts of data without the setting up of real sensors or need for volunteer subjects. Also, their annotations are more precise compared to real dataset methods (diary, smartphone apps, voice records).

Nevertheless, there are no synthetic datasets which are publicly available. Only some dataset generation tools such as OpenSH [4] are available to the public, but they require the user of the tool to maintain the simulation themselves. Moreover, these simulation environments often lack realism, due to a lack of variability, which does not allow the algorithms generalize well. In addition, virtual smart homes are not related to real smart homes. In general, these virtual smart homes are simplified versions of typical homes (small square homes, no floors), which also simplifies the selection, number and placement of sensors and the recording of sensor data.

In accordance with the dataset limitations described above, a human activity algorithm trained on a dataset corresponding to a particular configuration of house, inhabitants and life habits, will not be adapted to another house. From an industrial/commercial point of view, it is not feasible to ask the inhabitants to self-tag their activities for several months in order to collect enough data to train or adapt the final system. On the contrary, the system should be able to apply previously learned models and adapt them immediately to the new situation. However, generating data from simulation environments close to the final environment to adapt human activity recognition models, bridging the gap between training and test data, could be a solution.

2.9 Contribution Proposal

In this chapter, we have seen state of the art research on human activity recognition in smart homes based on home automation sensors through DDA. A significant amount of work has already been done. However, there are still many challenges to be met in order to allow these techniques to fully integrate into our daily lives and our homes.

In this thesis we focus on offline pre-segmented ADLs recognition and tackle the following:

- encoding, feature extraction, pattern recognition and relevant knowledge to better interpret sensor data;
- encode relevant time and previous context information;
- knowledge transfer from one home to another;
- data generation for application into the real world;

2.9.1 Encoding, Feature Extraction, Pattern Recognition and Relevant Knowledge

Just like words, sensor activations carry meaning thanks to their context and nature. They follow a certain syntax directly related to the syntax induced by the residents' actions. In other words, the syntax represents the logical sequence of the residents' actions during their activities. Moreover, similar to human activity recognition in smart homes, the NLP domain works on sequence analysis and classification problems.

This is why, in this work, in order to answer the first problem of feature extraction and context understanding, we propose to transfer methods used for coding and representation of words and language coming from the world of NLP to the domain of human activity recognition in smart homes. Indeed, in the NLP, many deep learning-based methods have been developed to encode, interpret, and understand words, texts, language, and sequences. These methods are able to understand semantics and syntax by making logical links and by understanding the various levels of dependency between words.

2.9.2 Time and Previous Context Information

Previous works consider human activities as sequences of sensor activations which reflect the actions of residents. They do not take into account temporal information. Time precision and understanding offers the possibility to create specific services or scenarios, as for instance, reminders for morning and evening medication intake. In addition, taking into account time information might improve the ADLs recognition.

The consideration of relations between activities is another interesting task. Throughout the day, we perform activities that are often linked. Being able to interpret these relations could help systems recognize the current activity, especially when similarities between sensor activation sequences lead to confusions.

2.9.3 Transfer Learning

Second, despite the ability of current human activity recognition models to automatically extract features and patterns to fit any dataset thanks to deep learning, these models remain fixed to the dataset on which they are trained. Once trained on the data of one house, it is not possible for them to work on another house. Moreover, it is not currently possible for an algorithm to take advantage of knowledge acquired on a larger dataset to generalize and increase its performance on a more precise dataset. The knowledge thus acquired to recognize ADLs on a dataset is not transferable. However, there would be an advantage in taking advantage of more general knowledge to improve performance in another context where the data are in smaller quantities. We therefore propose in this work to draw inspiration once again from the methods of the NLP world, where models are able to represent and store general knowledge in a latent space through embeddings. Thanks to these embeddings, the knowledge becomes transferable to other tasks, allowing the algorithms to train faster but also to reach higher performances on the final task they are intended for.

2.9.4 Data Generation

As we have seen over the past decades, more and more smart homes and real-world test beds are emerging. The current publicly available datasets from these real-world environments are not sufficient to cover the probability distribution of any possible activities, lifestyle or configuration variability and thus allow algorithms to generalize. These datasets are indeed limited in the number of activities and occupants they take into account, as well as the types of sensors and the type of resident.

Collecting new datasets is a very expensive and difficult task. The cost of sensors, the difficulty of accessing multiple types of homes, recruitment, collecting data from actual residents also takes time, and cannot be expedited. Moreover, the data requires labeling, i.e., segmentation of the data stream (start and end time of activities) as well as their class labels, which implies a significant investment on the part of the user and presents a high risk of error, as these annotations are performed manually and usually by the user himself.

In accordance with the dataset limitations described above, a human activity recognition algorithm trained on a dataset corresponding to a particular configuration of house, inhabitants and life habits, will not be adapted to another house. From an industrial/commercial point of view, it is not feasible to ask the inhabitants to self-tag their activities for several months in order

to collect enough data to train or adapt the final system. On the contrary, the system should be able to apply previously learned models and adapt them immediately to the new situation, based on only a few data from the test case: this is the principle of learning by transfer. It is therefore not necessary to have a large amount of data similar to the test case.

To address this challenge of the gap between the training data and the use case data, data generation could be a solution, thanks to the Digital Twin concept [31, 32]. In this thesis, we propose to bridge the gap between test and training data using a smart home simulator that can generate training data closer to the test cases by replicating real apartments in a Digital Twin and modeling complex ADLs in that Digital Twin.

Bibliography

[1] Fadi Al Machot, Heinrich C Mayr, and Suneth Ranasinghe. A windowing approach for activity recognition in sensor data streams. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 951–953. IEEE, 2016.

[2] Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. Aras human activity datasets in multiple homes with multiple residents. In *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 232–235. IEEE, 2013.

[3] Alaa Alhamoud, Vaidehi Muradi, Doreen Böhnstedt, and Ralf Steinmetz. Activity recognition in multi-user environments using techniques of multi-label classification. In *Proceedings of the 6th International Conference on the Internet of Things*, pages 15–23, 2016.

[4] Nasser Alshammari, Talal Alshammari, Mohamed Sedky, Justin Champion, and Carolin Bauer. Openshs: Open smart home simulator. *Sensors*, 17(5):1003, 2017.

[5] Samaneh Aminikhanghahi and Diane J Cook. Enhancing activity recognition using cpd-based activity segmentation. *Pervasive and Mobile Computing*, 53:75–89, 2019.

[6] Arni Ariani, Stephen J Redmond, David Chang, and Nigel H Lovell. Simulation of a smart home environment. In *2013 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering (ICICI-BME)*, pages 27–32. IEEE, 2013.

[7] Ibrahim Armac and Daniel Retkowitz. Simulation of smart environments. In *IEEE International Conference on Pervasive Services*, pages 257–266. IEEE, 2007.

[8] Parviz Asghari, Elnaz Soelimani, and Ehsan Nazerfard. Online human activity recognition employing hierarchical hidden markov models, 2019.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[11] Kevin Bouchard, Amir Ajroud, Bruno Bouchard, and Abdenour Bouzouane. Simact: a 3d open source smart home simulator for activity recognition. In *Advances in Computer Science and Information Technology*, pages 524–533. Springer, 2010.

[12] Alexis Brenon, François Portet, and Michel Vacher. Context feature learning through deep learning for adaptive context-aware decision making in the home. In *2018 14th International Conference on Intelligent Environments (IE)*, pages 32–39. IEEE, 2018.

[13] Eris Chinellato, David C Hogg, and Anthony G Cohn. Feature space analysis for human activity recognition in smart environments. In *2016 12th International Conference on Intelligent Environments (IE)*, pages 194–197. IEEE, 2016.

[14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[15] Diane Cook, Kyle Feuz, and Narayanan Krishnan. Transfer Learning for Activity Recognition: A Survey. *Knowledge and information systems*, 36:537–556, Sep 2013. doi: 10.1007/s10115-013-0665-3.

[16] Diane J Cook. Learning setting-generalized activity models for smart spaces. *IEEE intelligent systems*, 2010(99):1, 2010.

[17] Diane J Cook and Maureen Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480, 2009.

[18] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, 2012.

[19] Diane J Cook, Narayanan C Krishnan, and Parisa Rashidi. Activity discovery and activity recognition: A new partnership. *IEEE transactions on cybernetics*, 43(3):820–828, 2013.

[20] Julien Cumin, Grégoire Lefebvre, Fano Ramparany, and James L Crowley. A dataset of routine daily activities in an instrumented home. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 413–425. Springer, 2017.

[21] Emiro De-La-Hoz-Franco, Paola Ariza-Colpas, Javier Medina Quero, and Macarena Espinilla. Sensor-based datasets for human activity recognition–a systematic review of literature. *IEEE Access*, 6:59192–59210, 2018.

[22] Maxime Devanne, Panagiotis Papadakis, and Sao Mai Nguyen. Recognition of activities of daily living via hierarchical long-short term memory networks. In *International Conference*

on Systems Man and Cybernetics, pages 3318–3324. IEEE, Jul 2019. doi: 10.1109/SMC.2019.8914457.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[24] Macarena Espinilla, Javier Medina, and Chris Nugent. Ucami cup. analyzing the uja human activity recognition dataset of activities of daily living. *Proceedings*, 2(19):1267, Oct 2018. ISSN 2504-3900. doi: 10.3390/proceedings2191267. URL `http://dx.doi.org/10.3390/proceedings2191267`.

[25] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE international conference on big data (Big Data)*, pages 1367–1376. IEEE, 2018.

[26] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[27] Qiang Fu, Ping Li, Cai Chen, Lanlan Qi, Yongqiang Lu, and Chen Yu. A configurable context-aware simulator for smart home systems. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 39–44. IEEE, 2011.

[28] Alireza Ghods and Diane J Cook. Activity2vec: Learning adl embeddings from sensor data with a sequence-to-sequence model. *arXiv preprint arXiv:1907.05597*, 2019.

[29] Hristijan Gjoreski, Simon Kozina, Matjaz Gams, Mitja Lustrek, Juan Antonio Álvarez-García, Jin-Hyuk Hong, Julian Ramos, Anind K Dey, Maurizio Bocca, and Neal Patwari. Competitive live evaluations of activity-recognition systems. *IEEE Pervasive Computing*, 14(1):70–77, 2015.

[30] Munkhjargal Gochoo, Tan-Hsu Tan, Shing-Hong Liu, Fu-Rong Jean, Fady S Alnajjar, and Shih-Chia Huang. Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and dcnn. *IEEE journal of biomedical and health informatics*, 23(2):693–702, 2018.

[31] Michael Grieves. Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1:1–7, 2014.

[32] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, pages 85–113. Springer, 2017.

[33] Rebeen Ali Hamad, Alberto Salguero Hidalgo, Mohamed-Rafik Bouguelia, Macarena Espinilla Estevez, and Javier Medina Quero. Efficient activity recognition in smart homes using delayed fuzzy temporal windows on binary sensors. *IEEE journal of biomedical and health informatics*, 24(2):387–395, 2019.

[34] Rebeen Ali Hamad, Longzhi Yang, Wai Lok Woo, and Bo Wei. Joint learning of temporal models to handle imbalanced data for human activity recognition. *Applied Sciences*, 10(15): 5293, 2020.

[35] Rebeen Ali Hamad, Masashi Kimura, Longzhi Yang, Wai Lok Woo, and Bo Wei. Dilated causal convolution with multi-head self attention for sensor human activity recognition. *Neural Computing and Applications*, pages 1–18, 2021.

[36] S Helal, E Kim, and S Hossain. Scalable approaches to activity recognition research. In *Proceedings of the 8th international conference pervasive workshop*, pages 450–453, 2010.

[37] Sumi Helal, Jae Woong Lee, Shantonu Hossain, Eunju Kim, Hani Hagras, and Diane Cook. Persim-simulator for human activities in pervasive spaces. In *2011 Seventh International Conference on Intelligent Environments*, pages 192–199. IEEE, 2011.

[38] Brandon Ho, Dieter Vogts, and Janet Wesson. A smart home simulation tool to support the recognition of activities of daily living. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*, pages 1–10. 2019.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[40] 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(3):236 – 252, 2009. doi: https://doi.org/10.1016/j.pmcj.2008.05.002.

[41] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[43] Paula Lago, Fréderic Lang, Claudia Roncancio, Claudia Jiménez-Guarín, Radu Mateescu, and Nicolas Bonnefond. The contextact@ a4h real-life dataset of daily-living activities. In *International and interdisciplinary conference on modeling and using context*, pages 175–188. Springer, 2017.

[44] Jae Woong Lee, Seoungjae Cho, Sirui Liu, Kyungeun Cho, and Sumi Helal. Persim 3d: Context-driven simulation and modeling of human activities in smart spaces. *IEEE Transactions on Automation Science and Engineering*, 12(4):1243–1256, 2015.

[45] Xinyu Li, Yanyi Zhang, Jianyu Zhang, Shuhong Chen, Ivan Marsic, Richard A Farneth, and Randall S Burd. Concurrent activity recognition with multimodal cnn-lstm structure. *arXiv preprint arXiv:1702.01638*, 2017.

[46] Daniele Liciotti, Michele Bernardini, Luca Romeo, and Emanuele Frontoni. A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 04 2019. doi: 10.1016/j.neucom.2018.10.104.

[47] Javier Medina-Quero, Shuai Zhang, Chris Nugent, and Macarena Espinilla. Ensemble classifier of long short-term memory with fuzzy temporal windows on binary sensors for activity recognition. *Expert Systems with Applications*, 114:441–453, 2018.

[48] Andres Mendez-Vazquez, A Helal, and D Cook. Simulating events to generate synthetic data for pervasive spaces. In *Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*. Citeseer, 2009.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[50] Gadelhag Mohmed, Ahmad Lotfi, and Amir Pourabdollah. Employing a deep convolutional neural network for human activity recognition based on binary ambient sensor data. In *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 1–7, 2020.

[51] Anubhav Natani, Abhishek Sharma, and Thinagaran Perumal. Sequential neural networks for multi-resident activity recognition in ambient sensing smart homes. *Applied Intelligence*, pages 1–15, 2021.

[52] Fco Ordóñez, Paula De Toledo, Araceli Sanchis, et al. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13 (5):5460–5477, 2013.

[53] Bokyoung Park, Hyeongyu Min, Green Bang, and Ilju Ko. The user activity reasoning model in a virtual living space simulator. *International Journal of Software Engineering and Its Applications*, 9(6):53–62, 2015.

[54] Jiho Park, Kiyoung Jang, and Sung-Bong Yang. Deep neural networks for activity recognition with multi-sensor data in a smart home. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 155–160. IEEE, 2018.

[55] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[56] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[57] Matthai Philipose, Kenneth P Fishkin, Mike Perkowitz, Donald J Patterson, Dieter Fox, Henry Kautz, and Dirk Hahnel. Inferring activities from interactions with objects. *IEEE pervasive computing*, 3(4):50–57, 2004.

[58] Bronagh Quigley, Mark Donnelly, George Moore, and Leo Galway. A comparative analysis of windowing approaches in dense sensing environments. *Proceedings*, 2(19):1245, Oct 2018. ISSN 2504-3900. doi: 10.3390/proceedings2191245. URL `http://dx.doi.org/10.3390/proceedings2191245`.

[59] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[60] Muhammad Safyan, Zia Ul Qayyum, Sohail Sarwar, Raúl García-Castro, and Mehtab Ahmed. Ontology-driven semantic unified modelling for concurrent activity recognition (oscar). *Multimedia Tools and Applications*, 78(2):2073–2104, 2019.

[61] Mohamed SEDKY, Christopher HOWARD, Talal Alshammari, and Nasser Alshammari. Evaluating machine learning techniques for activity classification in smart home environments. *International Journal of Information Systems and Computer Sciences*, 12(2):48–54, 2018.

[62] Deepika Singh, Erinc Merdivan, Sten Hanke, Johannes Kropf, Matthieu Geist, and Andreas Holzinger. Convolutional and recurrent neural networks for activity recognition in smart environment. In *Towards integrative machine learning and knowledge extraction*, pages 194–205. Springer, 2017.

[63] Deepika Singh, Erinc Merdivan, Ismini Psychoula, Johannes Kropf, Sten Hanke, Matthieu Geist, and Andreas Holzinger. Human activity recognition using recurrent neural networks. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 267–274. Springer, 2017.

[64] Deepika Singh, Erinc Merdivan, Ismini Psychoula, Johannes Kropf, Sten Hanke, Matthieu Geist, and Andreas Holzinger. Human activity recognition using recurrent neural networks. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 267–274. Springer, 2017.

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

[66] Jonathan Synnott, Liming Chen, Chris D Nugent, and George Moore. The creation of simulated activity datasets using a graphical intelligent environment simulation tool. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4143–4146. IEEE, 2014.

[67] Jonathan Synnott, Chris Nugent, and Paul Jeffers. Simulation of smart home activity datasets. *Sensors*, 15(6):14162–14179, 2015.

[68] Tan-Hsu Tan, Munkhjargal Gochoo, Shih-Chia Huang, Yi-Hung Liu, Shing-Hong Liu, and Yun-Fa Huang. Multi-resident activity recognition in a smart home using rgb activity image and dcnn. *IEEE Sensors Journal*, 18(23):9718–9727, 2018.

[69] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing*, pages 158–175. Springer, 2004.

[70] Jawad Tayyub, Majd Hawasly, David C Hogg, and Anthony G Cohn. Learning hierarchical models of complex daily activities from annotated videos. In *IEEE Winter Conf. on Applications of Computer Vision*, pages 1633–1641, 2018.

[71] Son N Tran, Qing Zhang, Vanessa Smallbon, and Mohan Karunanithi. Multi-resident activity monitoring in smart homes: A case study. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 698–703. IEEE, 2018.

[72] Tim LM van Kasteren, Gwenn Englebienne, and Ben JA Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity recognition in pervasive intelligent environments*, pages 165–186. Springer, 2011.

[73] Timotheus Leonhard Martinus van Kasteren et al. Activity recognition for health monitoring elderly using temporal probabilistic models. ASCI, 2011.

[74] Aiguo Wang, Guilin Chen, Cuijuan Shang, Miaofei Zhang, and Li Liu. Human activity recognition in a smart home environment with stacked denoising autoencoders. In *International conference on web-age information management*, pages 29–40. Springer, 2016.

[75] Aiguo Wang, Shenghui Zhao, Chundi Zheng, Jing Yang, Guilin Chen, and Chih-Yung Chang. Activities of daily living recognition with binary environment sensors using deep learning: A comparative study. *IEEE Sensors Journal*, 21(4):5423–5433, 2020.

[76] LuKun Wang and RuYue Liu. Human activity recognition based on wearable sensor using hierarchical deep lstm networks. *Circuits, Systems, and Signal Processing*, 39(2):837–856, 2020. doi: 10.1007/s00034-019-01116-y.

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

[78] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016. doi: 10.1186/s40537-016-0043-6.

[79] Nawel Yala, Belkacem Fergani, and Anthony Fleury. Feature extraction for human activity recognition on streaming data. In *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6. IEEE, 2015.

[80] Surong Yan, Kwei-Jay Lin, Xiaolin Zheng, and Wenyu Zhang. Using latent knowledge to improve real-time activity recognition for smart iot. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

Chapter 3

Background, Approach and Experimental Methodology

3.1 Introduction

A significant amount of work has already been done for HAR in smart homes. Deep Learning techniques have shown exemplary improvement in this area, especially through the application of recurrent networks such as LSTM. However, there are still many improvements to envisage until these techniques become fully integrated into our daily lives and homes.

Despite the ability of LSTM to interpret temporal links and patterns within sequences of activity, classification scores are not always successful, especially when datasets become more complex. In the case of similar activities, in terms of sensor activations, when we deal with large numbers of activities or datasets containing the activities of several residents, LSTM-based approaches demonstrate great difficulties. Indeed, it seems that complementary knowledge and information must be extracted from the data to improve these models.

This model enrichment can be achieved by using new approaches to represent sensor events. The combination of FTW, to encode and extract features, and some LSTM showed good performances. The ability of FTW to encode more or less temporally distant sensor activations seems to be the key in such forms of segmentation and encoding methods. Nevertheless, the FTW can only be used with binary sensors although activations of smart homes sensors are not always binary, as temperature, humidity, CO2, etc. Techniques able to use multi-modal sensors are therefore needed. Moreover, FTW are built following predefined manual rules. It should be interesting to find an automatic way of avoiding their loss of generality.

Building models with a part for feature extraction (FTW) and a part for classification and pattern recognition (LSTM), seems to be a promising alternative. Mixing models adapted to certain sub-tasks, such as feature extraction or pattern extraction, could in fact be an opportunity for improvements. Classically, some parts of the model could be trained for a specific task, for example a first part for input coding and feature extraction, and a second part for the pattern recognition task from the previously generated features.

This chapter describes the ideas and the proposed approach likely to improve the recognition

algorithms of human activities in a smart home. Firstly, we detail the idea and the proposed approach through a step by step workflow for a better understanding. Secondly, We will describe the reference (i.e. baseline) models used for the experiments of this work. We will then present the datasets and the evaluation method used. Finally, we will try to explain the metrics that allow us to compare the proposed approach to the reference models.

3.2 Proposed Approach

3.2.1 Key Idea

This thesis proposes to consider the traces of different sensors as a single time series, in which the sensor activations follow each other, building, therefore, a sequence of sensor activations. This long sequence can then be divided into sub-sequences, each one corresponding to an activity. Within these sequences, the sensor activations define a pattern that allows to recognize the corresponding activity. This pattern is built up taking into account which sensor activations are present, their order of appearance and their frequency. The NLP domain, especially as far as the text or sentence classification tasks are concerned, also deals with sequences of words that define patterns. The motivations are for similar reasons with our concerns. Therefore, we can get fruitful inspiration from the language paradigm and draw technical advantages from the NLP domain methods to try to solve the HAR problem in smart homes. The goal by using this paradigm is to find means to represent and extract some information concerning part of the rationality of sensor activation relations. By analogy, just as language can describe activities with words, this work assumes that "the house has its own language to describe what is happening in it".

One could say that the activation of a sensor is similar to a word "spoken by the house" and a sequence of activations is similar to a sentence describing one or more activities. Sensor activations, like words in a sentence, are organized into a "syntax". Indeed, each sensor activation translates an operation or an action of a resident. Therefore, the order of the activation of the sensors (order of operations, order of actions), as well as the links and the relations between them, give information about the way the activity is performed. As some words can only appear after a particular one, sensor activations follow a similar logic. For example, the detection of movement in a room usually could occur after someone has opened or passed through a door. The opening of the faucet in the bathroom can only appear if a resident has entered the bathroom, and cannot appear following an entry into the living room. This syntax can then be used as a pattern to recognize.

Moreover, in the case where several residents live under the same roof, interpreting that certain orders of actions are possible or not, makes it easier to distinguish what each resident does. For example, let us consider two residents, each one performing a different activity. The first one cooks while the second takes a shower. In this case, the sensors of the house could

potentially capture the actions of each resident in the following order: entering the kitchen, entering the bathroom, opening the fridge door, turn on the bathroom light, turn on the kitchen faucet, turn on the hotplate, turn on the shower faucet, turn off the kitchen faucet, etc. It is therefore useful to be able to interpret the logic of the possible actions sequences.

In this work, the problem of recognizing human activities in a smart home is envisaged as a language modeling task and a text classification problem. Our approach consists of four parts. First, we segment the dataset into sub-sequences of activities using an EW. This step, turns the dataset into "sentences". Second, the sensor activations and, at the same time, the sequences of activations are encoded. Third, sensors sequences are turned into index in order to be usable by models. Fourth, a classifier, based on a neural network structure captures the patterns, as well as the best features, to classify the activation sequence into ADLs. In order to facilitate the understanding of our general approach and some of its details, we will follow the step by step workflow, described in Figure 3.1.

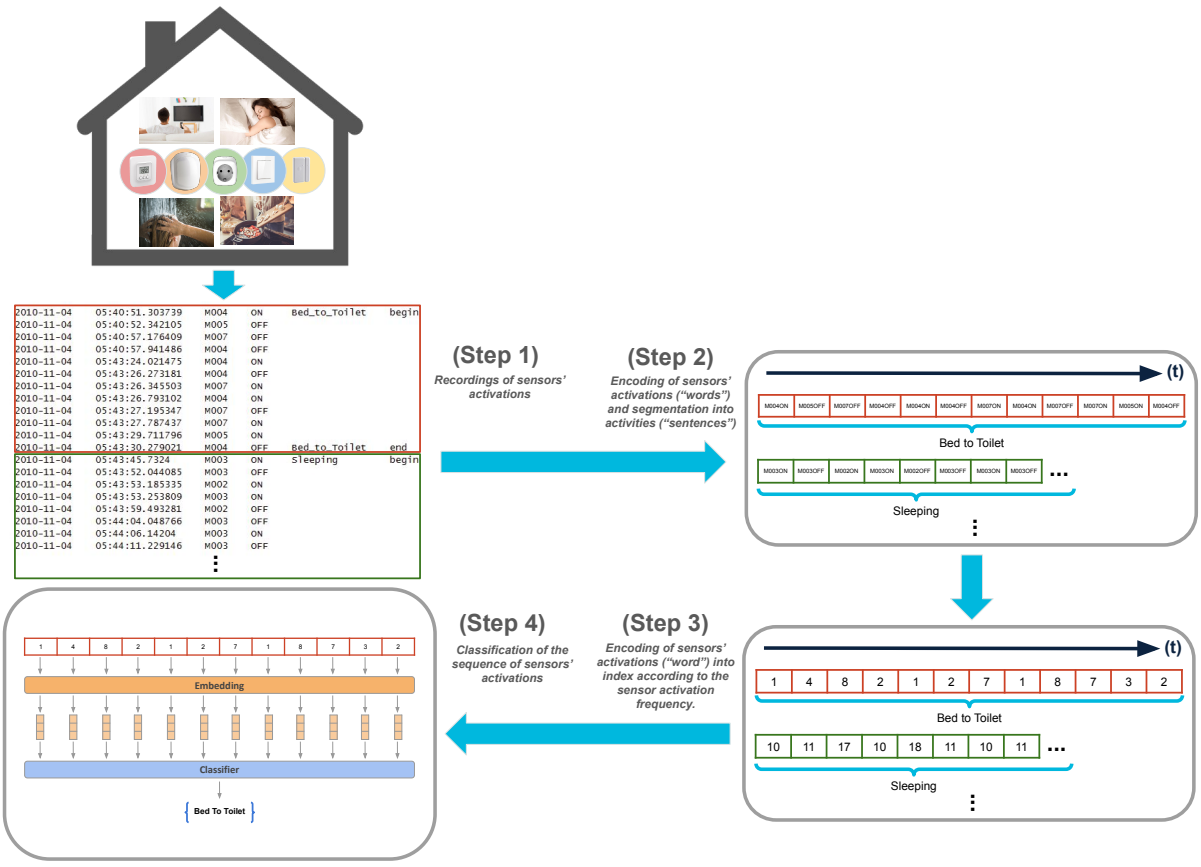


Figure 3.1 – Proposed model architecture and global workflow

3.2.2 Step 1 and Step 2: Sensors' Stream Segmentation and Encoding

From the stream of activations recorded by the sensors, *Step 1* performs a EW to segment the dataset into event sequences. Each sequence is an activity and the event timestamp order inside the sequences is kept. These event sequences are then encoded in *Step 2*.

The goal of the encoding is to obtain "words" that can be used as inputs of the model. To be able to represent all possible activations of the different sensors, we create a corpus of sensors' activations and consider these activations as categorical values, i.e. each sensor activation is represented by a unique value. By transforming the sensors' activations into categorical values, we allow the model to learn the notion of frequency of occurrence of an activation in a sequence. Moreover, the model, via such a representation, can capture the relation of an activation to another. All these categorical variables ("words") define the smart home vocabulary that describes activities.

More concretely, a set of sensors $S = \{s_1, ..., s_{|S|}\}$ produces events $e_i \in E$, E being the set of events. An event e_i is composed of the sensor ID s_i, the value v_i and the timestamp t_i. We concatenate the sensor ID s_i and its value v_i and ignore the timestamp t_i to create the "word" associated to this event, e.g., $s_i = M001$ and the binary value $v_i = ON$ becomes *M001ON*. For instance, the sequence of events for the activity *bed_to_toilet* in Figure 3.1 at *Step 1*, is turned into the sequence *[M004ON M005OFF M007OFF M004OFF M004ON M004OFF M007ON M004ON M007OFF M007ON M005ON M004OFF]*.

Sensors with numerical values are represented in the same way, as a concatenation of the sensor label and the measurement. For instance, if the activation of the temperature sensor named *T004* gives a value of 24.5, the input to the system is *T00424.5*.

3.2.3 Step 3: Words to Indexes

Each sensor activation in sequences is transformed into an index to be used as the input to a neural network. The index starts at 1 while the 0 value is reserved for the sequence padding. As in NLP domain, indexes are assigned following the "word" frequency (categorical value of the sensor activation frequency), e.g., if the "word" *M004ON* has the highest occurrence in the dataset, the assigned index is the lowest one, i.e. 1. This ordinal encoding encodes the frequency of a sensor activation. Thus, a sequence of "words" such as *[M005OFF M007OFF M004OFF M004ON M004OFF M007ON M004ON M007OFF M007ON M005ON M004OFF]* becomes the sequence of indexes *[1 4 8 2 1 2 7 1 8 7 3 2]*, according to the sensor activation frequency.

3.2.4 Step 4: Classification

Once the dataset is segmented into encoded sequences, these sequences are used by a neural network to perform classification. The model is composed of an embedding layer and a neural

network-based classification algorithm. The embedding is a projection layer that projects, in a d-dimensional space, each sensor activation. The role of embedding is to extract meaningful features that encode each sensor activation to help the classifier to perform its task. Thus, the classifier must extract from the embedding output, a list of meaningful vectors, patterns and features to recognize them. Via these patterns, it attributes the correct activity class label.

In this work, three types of classifiers have been studied: a CNN-based classifier and two LSTM-based classifiers. The CNN-based classifier is motivated by the strong ability of CNN to recognize patterns, while LSTM-based classifiers are inspired by the Liciotti et al. [6]'s work. Moreover, this work also studies advanced particular embedding methods inspired by the NLP domain to attempt to improve the ADL's recognition.

3.3 The Baseline Models

Thus, two of the currently best models in the field, defined by Liciotti et al. [6]'s work, are taken as baseline models in order to position our work. Liciotti et al. [6] study uses a similar sensors' sequence representation, as the one we advocate (sequential representation), and compare pitted several LSTM-based models against state-of-the-art machine learning methods (HMM, CRF, NB, etc) and CNN-based approaches. More in detail, their approach consists in encoding, under the form of categorical variables, the sequences of events of sensors before they pass them through the models. Liciotti et al. [6]'s work shows that models with the best classification results are the LSTM with embedding and the bidirectional LSTM with embedding (see Figure 3.2). These two models have shown exemplary results on five CASAS datasets (Milan, Cairo, Kyoto2 Kyoto3, Kyoto4)

3.4 The Experiments Datasets

This thesis work was conducted thought several experiments on three CASAS datasets: Aruba, Milan and Cairo, created by the Washington State University [3]. The data collected came from real houses equipped with smart sensors and with real inhabitants. The sensor events log, see example in Figure 3.3, are generated from motion sensors (sensor IDs begin with "M"), door closure sensors (sensor IDs begin with "D"), and temperature sensors (sensor IDs begin with "T"). The log is defined by one event by row and six columns that correspond respectively to the date of the event, the timestamp of the event, the ID of the sensor, the value taken by the sensor, the activity during which the sensor was triggered and finally the status of the activity (it's begin / it's end). The three selected datasets are different in terms of housing structure, see floor maps in Figure 3.4, number of inhabitants and contain several months of labeled activities, details in Table 3.1.

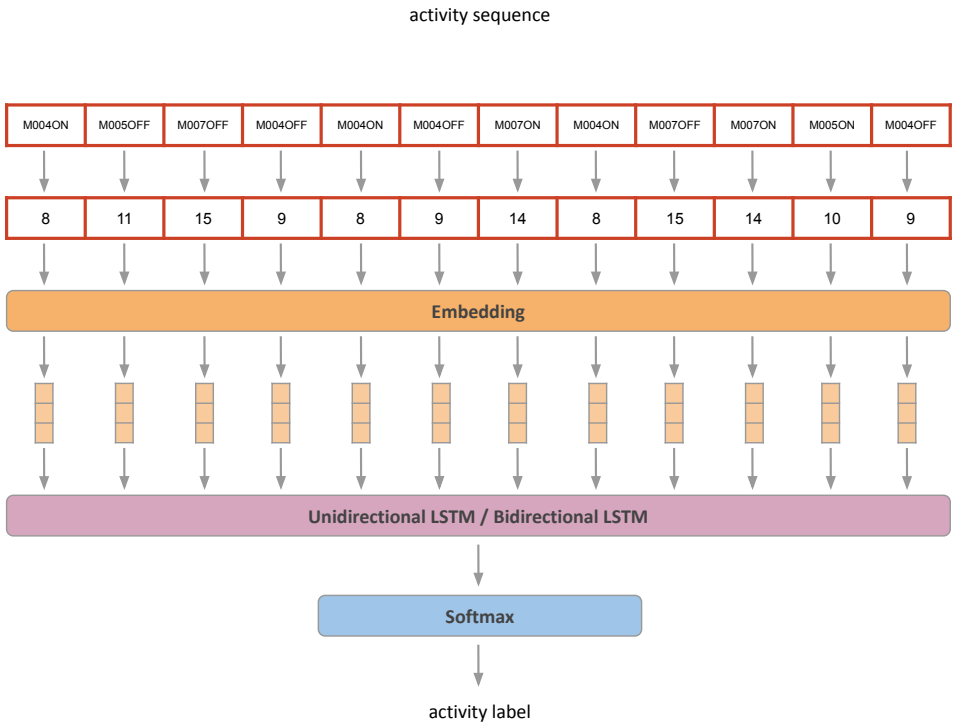


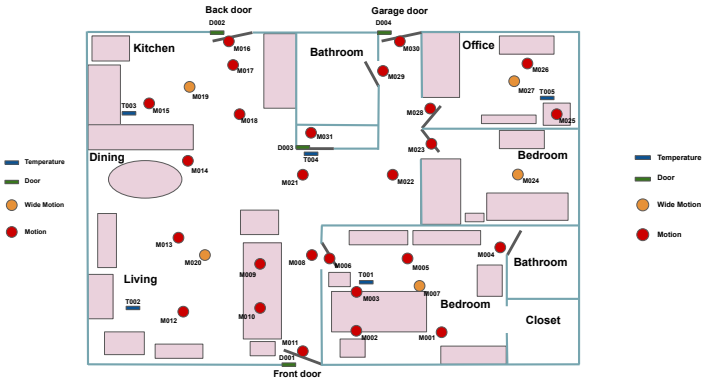
Figure 3.2 – Liciotti's baseline models

```
2010-11-04    05:40:51.303739    M004    ON     Bed_to_Toilet    begin
2010-11-04    05:40:52.342105    M005    OFF
2010-11-04    05:40:57.176409    M007    OFF
2010-11-04    05:40:57.941486    M004    OFF
2010-11-04    05:43:24.021475    M004    ON
2010-11-04    05:43:26.273181    M004    OFF
2010-11-04    05:43:26.345503    M007    ON
2010-11-04    05:43:26.793102    M004    ON
2010-11-04    05:43:27.195347    M007    OFF
2010-11-04    05:43:27.787437    M007    ON
2010-11-04    05:43:29.711796    M005    ON
2010-11-04    05:43:30.279021    M004    OFF    Bed_to_Toilet    end
2010-11-04    05:43:45.7324      M003    ON     Sleeping         begin
2010-11-04    05:43:52.044085    M003    OFF
2010-11-04    05:43:53.185335    M002    ON
2010-11-04    05:43:53.253809    M003    ON
2010-11-04    05:43:59.493281    M002    OFF
2010-11-04    05:44:04.048766    M003    OFF
2010-11-04    05:44:06.14204     M003    ON
2010-11-04    05:44:11.229146    M003    OFF
```

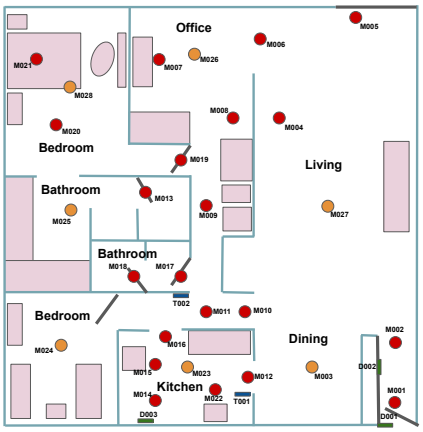
Figure 3.3 – Extract of a CASAS dataset log

Aruba and Milan are two houses with a single floor and have the same number of rooms but with a different disposition, while the Cairo house have three floors and more rooms than the two previous one. Aruba is a dataset of a woman living alone in a house, Milan contains the daily activities of a woman living with a dog and Cairo is a dataset of a couple living under the same roof with a dog.

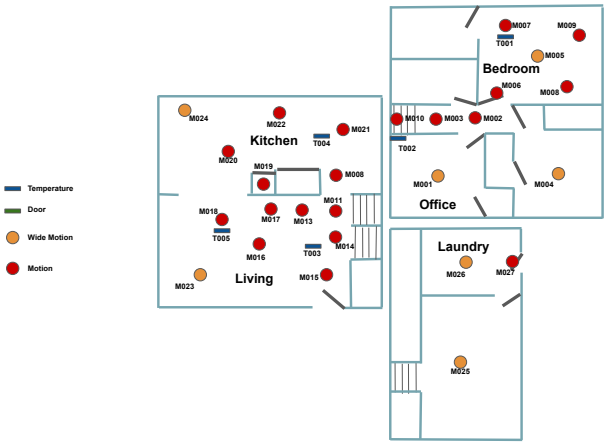
All the three datasets are unbalanced, meaning that some activities are less represented than



(a) Aruba dataset floor map



(b) Milan dataset floor map



(c) Cairo dataset floor map

Figure 3.4 – Floor maps of Aruba, Milan and Cairo Datasets.

Table 3.1 – Details des Datasets

	Aruba	Milan	Cairo
Residents	1	1+pet	2+pet
Number of sensors	39	33	27
Type of sensors	M,T,D	M,T,D	M,T
Number of activities	12	16	13
Number of days	219	82	56

others see, the Figures 3.5, 3.6, 3.7. This imbalance is explained by the fact that these datasets come from real homes with real inhabitants, and the activities are related to the lifestyle of these inhabitants. Moreover, as it can be seen on Figure 3.8 the three datasets doesn't cover

exactly the sames activities, it is therefore not easy to compare them directly. However, some of the activities could be grouped under a meta label or relabeled, in order to get a better comparison, e.g “Lunch”, “Dinner” and “Breakfast” from the Cairo dataset could become the activity “Cook”.

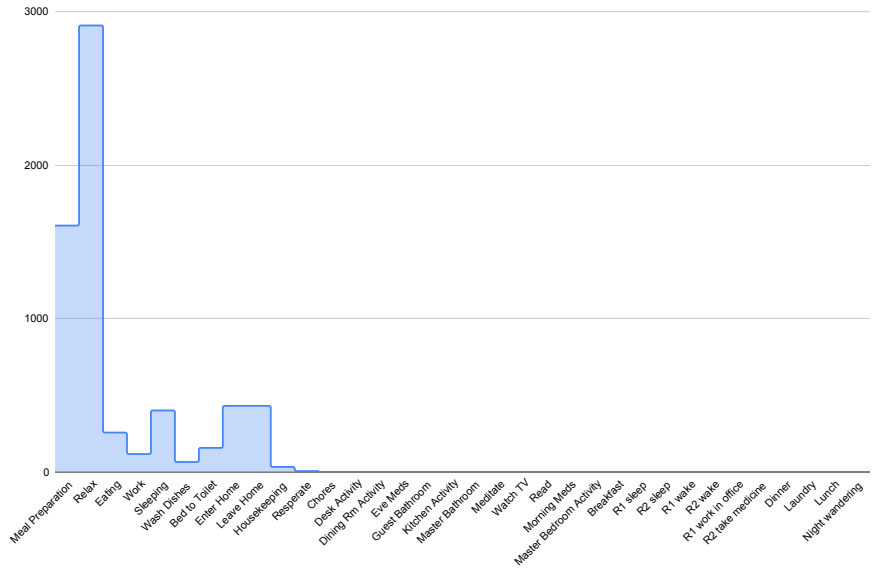


Figure 3.5 – Aruba’s activities labels and occurrences

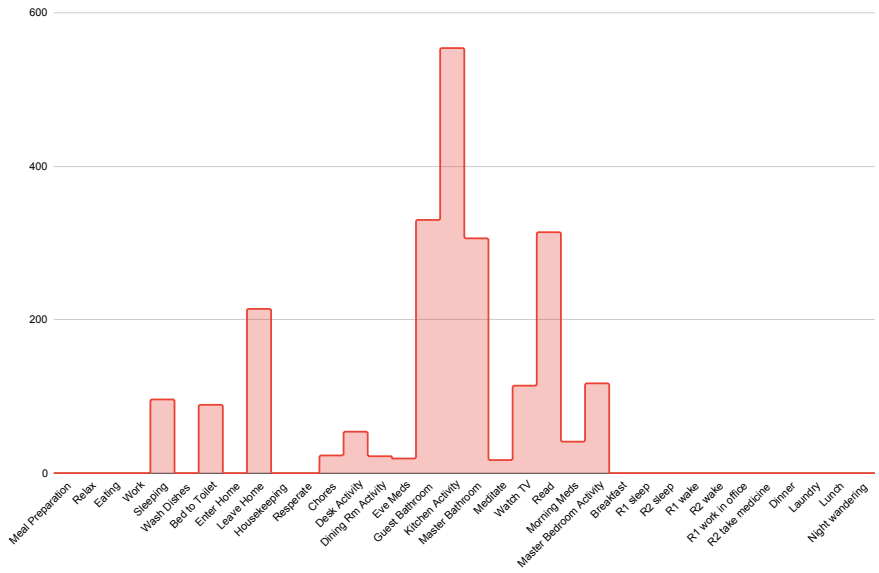


Figure 3.6 – Milan’s activities labels and occurrences

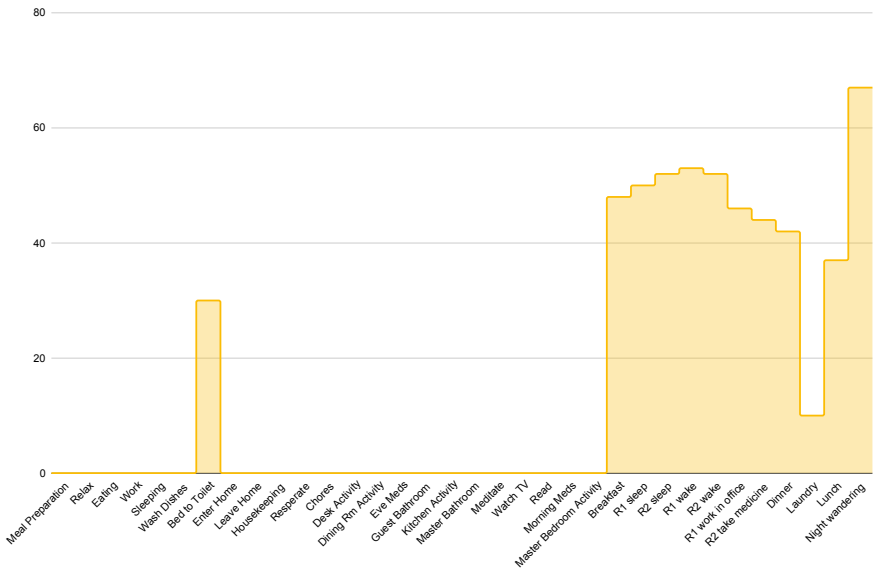


Figure 3.7 – Cairo's activities labels and occurrences

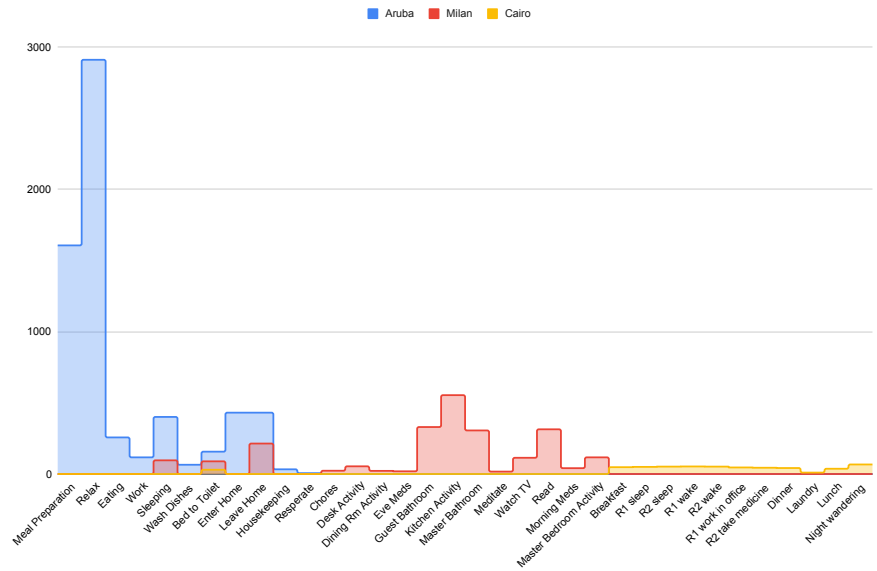


Figure 3.8 – Global occurrences and activities overlapping

3.5 Datasets Analysis and Pre-processing

3.5.1 Datasets Relabeling

Although each of the datasets originally included a large number of activities, for a part of this study we are interested in learning abstract models for 11 ADL activities as the works of Liciotti et al. [6] and Cook [2]. As seen before, a few number of activities are common in datasets, thus activities are grouped under new generic labels, detailed in the Table 3.2.

Table 3.2 – List of meta activities groups

	Milan	Cairo	Aruba
Bathing	Master Bathroom Guest Bathroom		
Bed to toilet	Bed to toilet	Bed to toilet	Bed to toilet
Cook	Kitchen Activity	Lunch Dinner Breakfast	Meal Preparation
Eat	Dining Room Activity		Eating
Enter home			Enter home
Leave home	Leave home	Leave Home	Leave Home
Personal hygiene			
Relax	Read Watch Tv		Relax
Sleep	Sleep	R1 sleep R2 sleep	Sleeping
Take medicine	Eve Meds Morning Meds	R2 take medecine	Resperate
Work	Desk Activity Chores	Laundry R1 work in office	Work Housekeeping Wash Dishes
Other	Meditate Master Bedroom Activity Other	Night wandering R2 wake R1 wake Other	Other

The purpose of this relabeling, according to the authors of [6] and [2], is to allow a fairer comparison between datastets. More than just allowing us to make a consistent comparison of the results, this relabeling allows us to be comparable with the previous cited works. The

choice of the new labels in works of Liciotti et al. [6] and Cook [2] is also motivated by the fact that these labels are typically used to discriminate the functional health of an individual within a clinical scenario [12]. The previous cited works have already grouped the activity for the Milan and Cairo datasets, thus we used the same categorization for these two datasets and have grouped by our own the activities labels of the Aruba dataset.

It should be noted that this relabeling also has an effect of partially rebalancing the datasets, but also, paradoxically, increases the number of examples of a particular class called "Other". This class corresponds to unidentified sensor activations or unidentified activation sequences, i.e. sensors events between the end of an activity and the beginning of another. As this class often represents more than 50% of the dataset, it introduces a bias which should be emphasized and kept in mind when analyzing the results. For example, if the classification algorithm is able to classify correctly all of the items in this class as "Other", then the global accuracy of this algorithm will be formally reported to be at least 50%, minimizing its actual accuracy for classes with meaningful labels.

3.5.2 Anomaly cleaning of datasets

After a detailed analysis of the datasets, we noticed that they can contain anomalies, especially in the "Milan" dataset: (1) they may contain duplicated data, full or partial days; (2) some row in the file contains characters errors, e.g "Oc" instead a "ON" value; (3) sensor activation traces in the dataset are not correctly ordered temporarily, i.e., within the time series, events with a later timestamp may be recorded before events with an earlier timestamp.

Activities in datasets are labeled with a "start" or "end" keyword to set when an activity begins and when it ends. However, activities can be nested, i.e., start1 start2 end2 end1. Moreover, it appends that they are interleaved, i.e., start1 start2 end1 end2. According to these observations, it is therefore important to pay attention to these particular cases when pre-segmenting the dataset into sequences of activities or to associate a label to a sensor event.

We observed, by reproducing the work of Liciotti et al. [6], that this cleaning has clearly an impact on the final results. We have noticed a loss of 5 points of accuracy on the "Milan" dataset using the bi-directional LSTM model proposed by authors. This loss can be explained in particular by a decrease in the number of occurrences of the "Other" class.

3.6 The Training and Evaluation Method

3.6.1 The Stratified K-fold Cross Validation

In this thesis, the stratified K-fold cross validation method [7] was chosen to evaluate and compare our proposed approach against the two selected baseline models. In particular, we

used for our experiments the stratified (over class) threefold cross-validation procedure whith shuffling, see Figure 3.9.

This choice was motivated by our intention to apply the evaluation strategy used in Liciotti et al. [6]'s work. Moreover, it seems to be the best strategy as we work with pre-segmented activity sequences. Indeed, we want to recognise and classify sequence of activities independently. In addition, to help models to be more general, it is better to shuffle all sequences and avoid time drift problem (if an activity is performed differently as the dataset progresses).

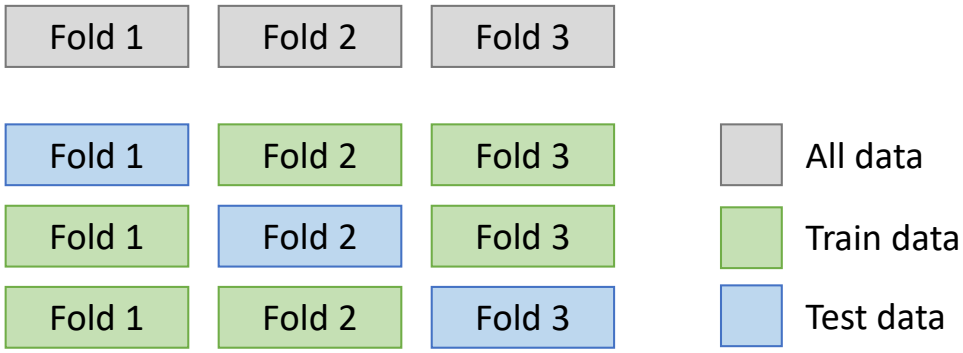


Figure 3.9 – K-fold cross validation principle.

The main idea of this evaluation method is to first shuffle all data (activity sequences) and then split them into, here, three folds. The split is applied in a so-called "stratified" manner, whose goal is to make folds by preserving the percentage of samples for each class. In other words, each fold contains the same percentage of samples for each class. Finally, each fold is used as the test set, while the two others are used for training.

3.6.2 The Process

To perform the training, the validation and the test of algorithms, we apply the procedure detailed in Figure 3.10. First, one fold is reserved for the final test (33% of the dataset). Then the two other folds are concatenated, shuffled and 20 % is kept for the validation step. These 20% validation data are not considered for training the model. The parameters' model are trained with the 80% data left thanks to the sparse categorical cross-entropy loss.

At the end of each training epoch, when all training data was seen by the model, the trained model is validated on the 20% reserved data. This step allows us to observe the model performance thanks to the validation loss and to check if it is starting to overfit (the model memorizes the training data).

During the training, we save and keep only the best model. To select the best model at each epoch, we check if the current model has a better accuracy than the previous one (the one at the previous epoch). If it is the case, the model is saved, otherwise a new epoch of training begins.

In order to stop the model training and avoid overfitting, we used a method so-called "early stop". It consists in stopping the model training if the validation loss doesn't decrease since a specified number of epochs.

When the training is finished, the "best" model is finally evaluated with the test fold. Results are saved and all the processes restart with another fold as test set. At the end, the reported results are the average of scores obtained on test sets.

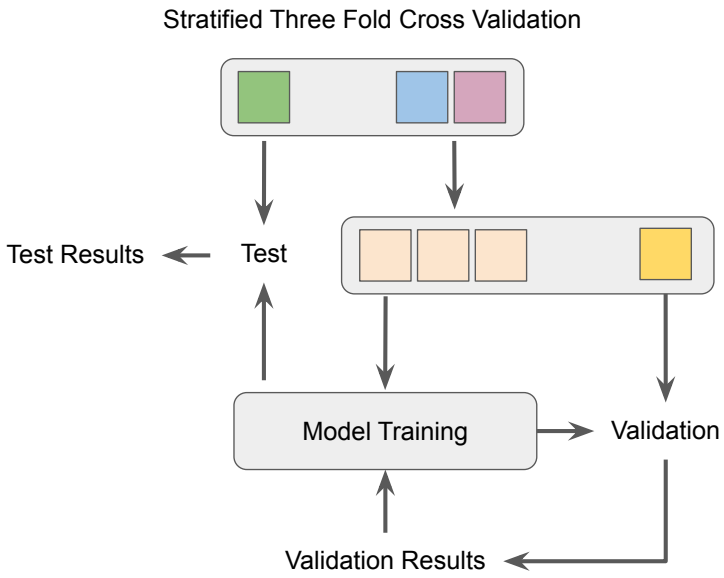


Figure 3.10 – Classification Stage

3.7 Metrics

3.7.1 Background

Using a meaningful measure of generalizability is an essential requirement for evaluating the performance of a classification algorithm on a given dataset. Since the actual ability of an algorithm to correctly predict class labels of unseen data could only be determined if an infinite amount of test data were available, generalizability must be approximated by an estimate. The ADLs recognition problem is a multi-class classification problem. The researchers have designed and used different metrics for classification tasks [11].

In the literature on recognition of ADLs metrics such as "Accuracy", "Precision", "Recall", and "F-Score" are mainly used to evaluate the algorithms [8, 10, 47]. These metrics are defined using four features, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The F-score, also called the F1-score, is a measure of the precision of a model on a dataset. The F-score is a way of combining the precision and recall of the model, and it is

defined as the harmonic mean of the precision and recall of the model. This metric is often the most relevant one to evaluate a model.

These measures are fine for balanced datasets, but real house datasets for recognition of ADLs are generally unbalanced. In other words, some activities have more examples than others. In an unbalanced data set, a minority class is more difficult to predict because there are few examples of that class, by definition. It is more difficult for a model to learn the characteristics of minority classes, but also to differentiate them from majority classes. Thus, it is important to evaluate whether a method or an algorithm is able to recognize this type of class or not. This type of evaluation demonstrates the strength of the method to not only look for features that separate the majority classes from the others, but features that seek to discriminate all classes. Therefore, it would be more appropriate to use measures able to deal with unbalanced datasets to properly evaluate the performance of the algorithms [4, 5].

3.7.2 Balanced Accuracy

The Balanced Accuracy [1] expressed by Equation 3.1, avoids inflated performance estimates on imbalanced datasets. It is the macro-average of Recall scores per class or, equivalently, raw accuracy where each sample is weighted according to the inverse prevalence of its true class. Thus, for balanced datasets, the score is equal to accuracy.

$$Balanced\_Accuracy = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}) \tag{3.1}$$

3.7.3 Weighted Metrics

The "macro-average" scores simply calculates the mean of the metrics, giving equal weight to each class. In problems where infrequent classes are nonetheless important, macro-averaging is a means of highlighting their performance. The "weighted" accounts for class imbalance by computing the average of metrics in which each class's score is weighted by its presence in the true data sample. Weighted measure allows us to validate if a model focuses on majorities classes or not. A high weighted score and low macro-average average score for the same metric means the model focuses on most representative classes and vice versa. Weighted Precision, Weighted Recall and Weighted F1-score are defined by Equations 3.2, 3.3, 3.4 respectively

$$Weighted\_Precision = \frac{Precision_{c1} * support_{c1} + ... + Precision_{c_n} * support_{c_n}}{Total\_support} \tag{3.2}$$

$$Weighted\_Recall = \frac{Recall_{c1} * support_{c1} + ... + Recall_{c_n} * support_{c_n}}{Total\_support} \tag{3.3}$$

$$Weighted\_F1\_score = \frac{F1\_score_{c1} * support_{c1} + ... + F1\_score_{c_n} * support_{c_n}}{Total\_support} \tag{3.4}$$

3.7.4 Library and Report Tools

In order to evaluate and compare the performances of the algorithms, we have chosen to use the "classification_report" method provided by the "scikit-learn" library [9]. This method allows us to generate detailed reports that contain all the previous mentioned metrics excepted the Balanced Accuracy.

However, as complement to the measures provided by the report function, we used the Balanced Accuracy computation method also provided by the "scikit-learn" library. Thanks to all these metrics, we will be able to compare more finely the proposed methods to other algorithms and highlight their performances or their handicap, in particular for the recognition of minority classes.

3.8 Summary

This chapter introduced the approach developed in this thesis to attempt to improve ADLs recognition algorithms. The proposed approach is inspired by the language paradigm in order to better understand sensors' activation links, relations and meanings. The main hypothesis is, such as words in a sentence, sensor's activation in an activity observes a particular syntax, meaning and context according to the performed activity. Understanding this kind of information and knowledge should help to improve ADLs recognition algorithms. This chapter also describes the two Liciotti et al. [6]'s baseline models used for all experiments. It introduced the datasets exploited for the different studies and explained their prepossessing. It also detailed the evaluation methodology and the metrics used in the following chapters.

The next chapter addresses pattern recognition issues with respect to activity recognition. An evaluation of a first approach using a CNN-based classifier as pattern recognition algorithm combined with an embedding as automatic features extraction algorithm is provided.

Bibliography

[1] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[2] Diane J Cook. Learning setting-generalized activity models for smart spaces. *IEEE intelligent systems*, 2010(99):1, 2010.

[3] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, 2012.

[4] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*, volume 10. Springer, 2018.

[5] Haibo He and Yunqian Ma. Imbalanced learning: foundations, algorithms, and applications. 2013.

[6] Daniele Liciotti, Michele Bernardini, Luca Romeo, and Emanuele Frontoni. A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 04 2019. doi: 10.1016/j.neucom.2018.10.104.

[7] Matthew D Mullin and Rahul Sukthankar. Complete cross-validation for nearest neighbor classifiers. In *ICML*, pages 639–646. Citeseer, 2000.

[8] Jiho Park, Kiyoung Jang, and Sung-Bong Yang. Deep neural networks for activity recognition with multi-sensor data in a smart home. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 155–160. IEEE, 2018.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Deepika Singh, Erinc Merdivan, Sten Hanke, Johannes Kropf, Matthieu Geist, and Andreas Holzinger. Convolutional and recurrent neural networks for activity recognition in smart environment. In *Towards integrative machine learning and knowledge extraction*, pages 194–205. Springer, 2017.

[11] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[12] Virginia G Wadley, Ozioma Okonkwo, Michael Crowe, and Lesley A Ross-Meadows. Mild cognitive impairment and everyday function: evidence of reduced speed in performing instrumental activities of daily living. *The American Journal of Geriatric Psychiatry*, 16(5): 416–424, 2008.

Chapter 4

The Time Series Classification Approach

4.1 Introduction

During the last two decades, Time Series Classification (TSC) has been considered one of the most challenging problems in data mining [3]. Due to its natural temporal ordering, time series data is present in almost every task that requires human cognitive processing [6]. Any classification problem using data registered by taking into account some notion of ordering can be cast as a TSC problem [4]. Time series are encountered in many real-world applications such as electronic health records [12], acoustic scene classification [10], cybersecurity [15] and in particular HAR [11, 18].

In smart homes, human actions during activities are in fact logged by sensors as state changes along minutes, hours, and days and then ordered by the system. In other words, these logs contains sequences of state changes along time that translate the ADLs. Thus, the ADLs recognition problem can be modeled as can be modeled as a TSC problem. Through this modeling, the objective is to extract significant characteristics and sufficiently descriptive patterns from a time series.

Recently Wang et al. [19] provide a standard baseline to exploit deep neural networks for end-to-end TSC without any crafting in feature engineering or data pre-processing. They proposed in their work an adaptation of two models from the field of computer vision, the Residual Networks (ResNet) and the Fully Convolutional Network (FCN). They showed that through pure end-to-end training of the raw time series data, both models achieve the best performance compared to other state-of-the-art models. In particular, the FCN wins 18 out of 44 problems and is in the top 5 of the other 26. Two years later Fawaz et al. [3] determined via a comparative study of state-of-the-art TSC algorithms with more datasets (96 datasets) that FCN could also achieve good results on many of the datasets they used. These works show that the FCN has great capabilities for feature extraction and classification.

However, even if FCN has great capabilities for feature extraction, works by Tahir et al. [16] and Yan et al. [21], have shown the importance of good feature representation for HAR

applications. Designing these features is a tedious task. Through deep learning algorithms and to avoid manual features representation, researchers in the NLP field have devised many useful algorithms and methods such as embeddings to represent word meanings and model language to increase the performance of deep learning classifiers.

This chapter presents a deep learning methodology for recognizing ADLs inspired by the NLP and TSC domains. This approach combines the projection of the input in an embedding with the FCN classifier algorithm. The embedding layer extracts a level of features and representations from the sensors, while the FCN layer performs pattern extraction with another level of extracted features and performs the final classification. The objective of this approach is to address the challenge of pattern recognition and classification generated by the classification of ADLs.

4.2 The Fully Convolutional Network

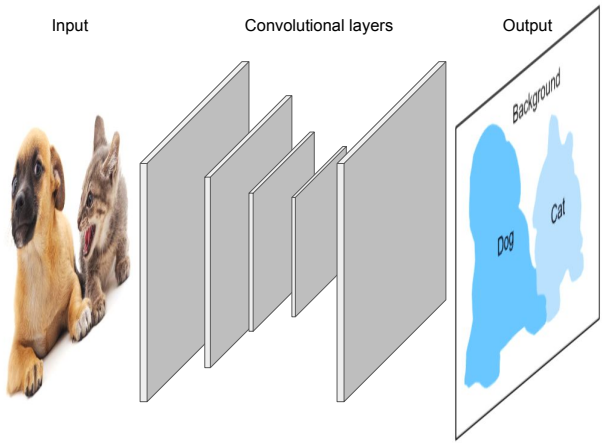


Figure 4.1 – Fully Convolutional Network (FCN) for segmentation

Originally, the FCN was designed to aid in semantic image segmentation [9], where it has shown compelling quality and efficiency at this task. The main idea of this model is to consider each output pixel as a classifier corresponding to the receptive field, and thus the network can be trained pixel by pixel according to the semantic segmentation annotation by category, see Figure 4.1. The FCN is a convolutional network that does not contain any local pooling layers or fully connected layers but, uses a convolutional layer as the final layer. Due to its performance on feature extraction, Wang et al. [19] transferred the FCN to TSC problems.

The architecture proposed by Wang et al. [19], see Figure 4.2, follows the original architecture almost exactly. One of the main characteristics of this architecture is the replacement of the final fully connected layer, found in traditional convolutional networks, and the convolutional layer of the original architecture by a Global Average Pooling (GAP) layer [8] and a softmax layer.

The GAP layer drastically reduces the number of parameters while enabling the use of the Class Activation Map (CAM), as proposed by Zhou et al. [22]. The CAM is a method that highlights which parts of the input contributed the most to a certain classification.

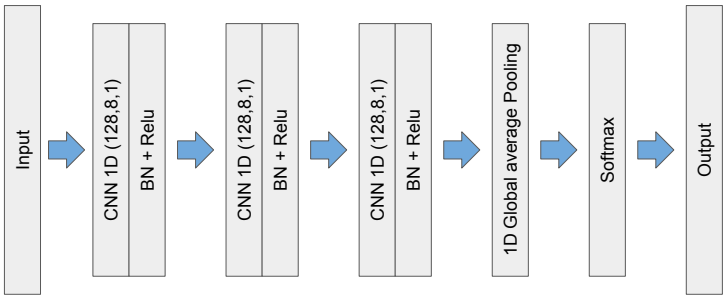


Figure 4.2 – Fully Convolutional Network (FCN) for Time Series Classification (TSC)

In greater detail, the architecture is composed of three blocks described by EQ 4.1. Where x is the input, W the weight matrix, b the bias, $\otimes$ the convolution operator, and h the hidden representation. Each block consists of a 1D convolutional layer with Batch Normalization (BN) [5] and a Rectified Linear Unit (ReLU) activation to speed up the convergence and help improve generalizations. After the three convolution blocks, features are fed into the GAP layer. The idea is to generate one feature map for each corresponding category of the classification task. The resulting vector is fed directly into the softmax layer to create the final classification.

$$\begin{aligned} y &= W \otimes x + b \\ z &= BN(y) \\ h &= ReLU(z) \end{aligned} \tag{4.1}$$

One advantage of GAP over the use of fully connected layers is that GAP is more native to the convolution structure thanks to how it enforces correspondences between feature maps and categories. Thus, the feature maps can be easily interpreted as category confidence maps. Another advantage is that there is no parameter to optimize in the GAP; thus, over fitting is avoided at this layer. Furthermore, GAP sums out the spatial information, which makes it more robust to spatial translations of the input. Meanwhile, the main advantage of FCN is the invariance in the number of parameters across time series of different lengths. This invariance due to using a GAP layer enables the use of a transfer learning approach, where one can train a model on a certain source dataset and fine-tune it on the target dataset [2].

4.3 The Fully Convolutional Network for Activity of Daily Lining Recognition in Smart Homes

4.3.1 The Model

To increase the FCN performance, we propose to add an embedding layer between the input and the FCN classifier, see Figure 4.3. In this method, as explained in the previous chapter, we encode the input sensors' activation sequences into sequences of categorical variables which represent the sensors' activation sequences. Then, we project these sequences into the embedding to extract the meaningful features. The embedding turns the univariate time series into a multivariate time series where each channel is a feature. The FCN then uses the embedding output to extract patterns and new features to classify each sequence.

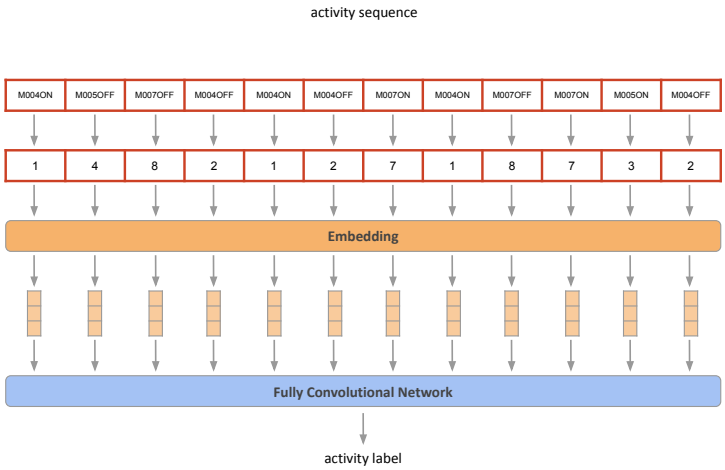


Figure 4.3 – Fully Convolutional Network (FCN) for Activity of Daily Lining (ADL) recognition

4.3.2 The Evaluation

In order to evaluate the method, the unidirectional LSTM and our FCN were compared with the widely used CASAS [1] benchmark datasets, describe earlier (cf. chapter 3), for two experiments. The first experiment evaluates: I. the abilities of both models to extract patterns and features from pieces of activity sequences; II. the contribution of the use of an embedding for the classification of ADLs; III. compare training times. The second experiment is a comparison between both models using the full activity sequences for classification instead of sequence pieces. It evaluates the performance of both algorithms to use short and long dependencies and patterns to recognize the ADLs.

4.4 Pattern Recognition Evaluation

4.4.1 Method

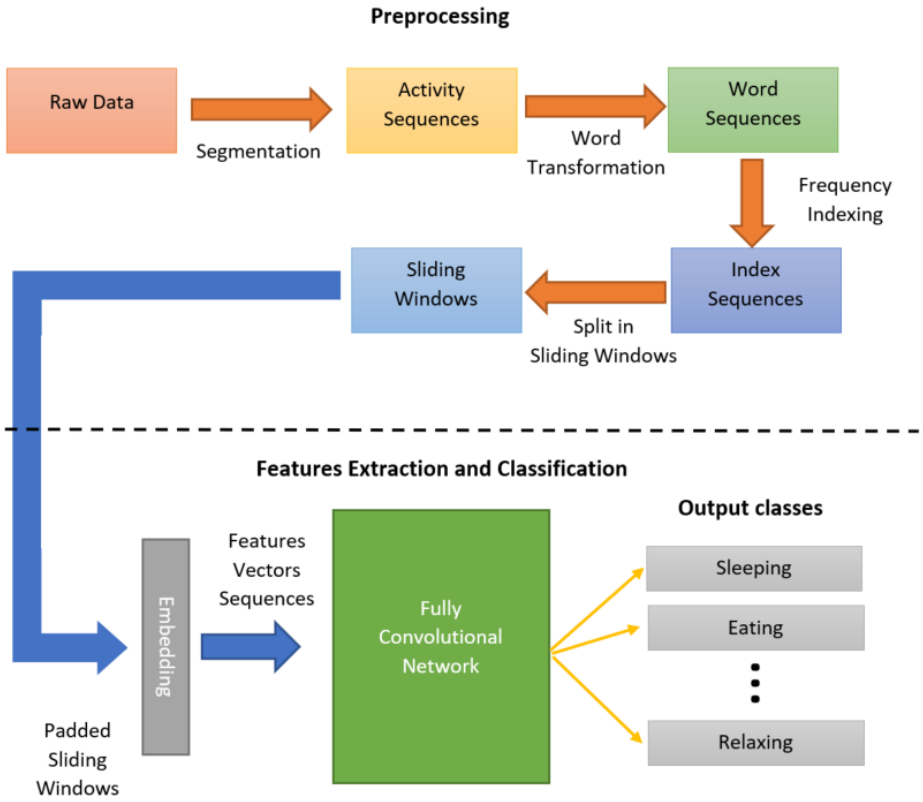


Figure 4.4 – Fully Convolutional Network (FCN) and SEW framework

In this experimentation, we evaluate the abilities of both models to extract patterns and features. To conduct this evaluation, we followed the framework presented in Figure 4.4. The input of our models is a sliding window of SEW that contain a piece of an encoded activity sequence. Depending on the SEW size and the activity length, some windows are padded with zero values to maintain a constant input length, this process is called “zero padding”. Once all the activity sequences have been cut into smaller windows and annotated with the label of the activity from which it originates, we distribute them into subsets for training and testing the algorithms. This distribution is done using the Stratified 3-fold cross validation method detailed in chapter 3.

4.4.2 FCN and LSTM Comparison: Pattern Recognition Abilities

Table 4.1 show the performances of two FCN and two LSTM on raw sensor data for the two datasets (Aruba, Milan). Vanilla unidirectional LSTM / FCN and unidirectional LSTM / FCN with an embedding layer on different window sizes were evaluated. The average balanced accuracy and the average F1-score of were computed from the 3-fold cross validation results. FCN appears to obtain the best F1-score with and without the embedding applied to both datasets. The LSTM is close to or equal to the FCN on a large SEW (windows greater than 50 sensors' activations). Compared to the FCN the LSTM seems to need more events to generate the classification.

From a balanced accuracy point of view, FCN often get the best values. Except when using the Milan dataset with a window size higher than 50. This decrease in performance seems due to the zero padding. Indeed, the average sequence length on the Milan dataset is around 88 events. When the window size is similar or greater than this average, the performance of the FCN decreases. Some small sequences like "Bed to Toilet" are so not classified. This results in a drop in the balanced accuracy score.

The FCN obtained the best values with SEW of sizes 50 and 25. Performance decreased as the SEW size decreased, but the FCN maintained a high score for balanced accuracy and the F1-score. Performance dropped less with FCN than with LSTM. It seems that FCN can generate more relevant automatic features than LSTM when using smaller sequences which contain less information. Moreover, look at results on both datasets, using an embedding layer increase the classification performances of both models. The embedding layer is thus, a relevant layer to extract meaningful features for both models and should be more studied.

Table 4.1 – F1 Score and Balanced Accuracy for Aruba and Milan datasets

	Aruba				Milan			
Model	100	75	50	25	100	75	50	25
	F1 Score (%)							
LSTM	96.67	94.67	90.67	85.00	84.00	85.67	75.33	64.00
FCN	99.00	98.00	97.67	92.33	77.33	93.67	88.33	83.67
LSTM + Embedding	**100.00**	99.67	98.00	90.00	98.00	97.00	93.00	73.67
FCN + Embedding	**100.00**	**100.00**	**100.00**	**99.00**	**99.00**	**98.00**	**97.00**	**94.33**
	Balanced Accuracy (%)							
LSTM	81.45	76.09	71.05	83.30	62.15	64.95	55.70	43.29
FCN	88.85	87.41	87.08	80.32	42.24	76.41	71.82	71.34
LSTM + Embedding	94.55	93.61	90.20	74.81	**88.52**	**86.77**	82.05	59.35
FCN + Embedding	**95.37**	**95.07**	**94.89**	**92.44**	84.23	86.64	**87.83**	**90.86**

4.4.3 FCN and LSTM Comparison: Training Time

Table 4.2 show the average training time and the average amount of training epochs by SEW size. On both datasets, FCN had the shortest times for every SEW size. The embedding layer allowed the system to reduce the number of epochs and the total training time in the majority of cases. The training time was divided by 2 to 6, with the FCN depending on the window size compared to LSTM. This time saving is explained by the ease of parallelization of calculations of convolutional networks.

Table 4.2 – Training time performance and number of epochs training for Aruba and Milan dataset

	Aruba				Milan			
Model	100	75	50	25	100	75	50	25
	Average epoch number							
LSTM	242	278	335	256	274	385	365	324
FCN	77	71	111	108	45	101	87	145
LSTM + Embedding	161	191	210	161	255	290	320	183
FCN + Embedding	67	62	71	98	65	51	52	55
	Average training time (HH:MM:SS)							
LSTM	06:28:42	06:43:08	06:29:58	03:00:26	02:03:43	02:11:07	01:44:06	01:00:10
FCN	00:58:00	00:52:15	01:20:35	00:51:27	00:09:39	00:20:17	00:15:08	00:16:50
LSTM + Embedding	04:45:56	04:45:38	04:14:35	02:02:53	01:57:52	01:49:35	01:36:56	00:35:26
FCN + Embedding	01:12:37	00:59:42	00:57:27	00:52:15	00:16:24	00:11:42	00:10:00	00:07:67

4.5 Temporal Dependencies Evaluation

4.5.1 Method

Previous experiments show that, compared to the LSTM, FCN is faster to train and seems better at extracting and recognizing patterns in SEWs. Moreover, the experiments show that the use of an embedding layer increases the performance of both models. Nevertheless, it also shows that when the SEW is padded with zeros, in the case the SEW is larger than the activity sequence, the FCN loses performance. This suggests that the FCN is experiencing padding difficulties.

The previous experiment was performed using small windows, which does not allow us to see whether FCN can interpret temporal dependencies between sensor events like LSTM. In order to evaluate this, we will use the FCN with embedding model and the bidirectional LSTM, directly on the EWs. In other words, we now will use the entire activity sequence as input. Thus, the models will have access to the whole sequence, thereby allowing them to evaluate the whole activity. In doing so, they can extract patterns and interpret the temporal dependencies within the sequence. This information is important because some events that are more or less

spaced temporally in a sequence can be correlated or even dependent. Which thus allows for better discrimination of the sequences between them. On contrary to the previous experiment, we used the bidirectional LSTM with embedding proposed in Liciotti et al. [7]'s work and not the unidirectional LSTM. This choice is motivated by the fact that the input is now longer, but also because this model has shown the best performance on the sequence classification problem for HAR in smart homes.

Some activities last longer than others, or contain more or fewer actions than others. Therefore, the sequences captured by sensor activations are often of different sizes, as they can contain greater or fewer sensor events. To train the algorithms, it is necessary for the sequences to be the same size. Sequences are either truncated, because they are too long for memory / learning efficiency reasons, or filled in with zeros, because they are too short.

To train the algorithms, as before, we used the Stratified 3-fold cross validation strategy (see chapter 3). Here, all EWs are distributed in three sets of data randomly but with equal distribution in terms of class.

4.5.2 FCN and LSTM Comparison: padding trouble

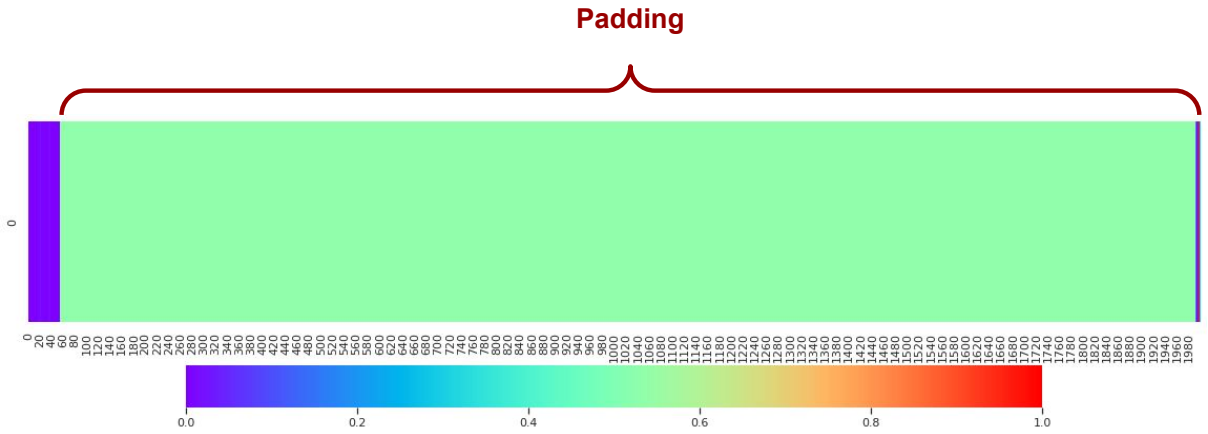
Table 4.3 – Comparison between embedding + bidirectional LSTM and embedding + FCN with zero padding

	embedding + bidirectional LSTM			embedding + FCN		
	Aruba	Milan	Cairo	Aruba	Milan	Cairo
Accuracy	**96.60**	**91.47**	**85.32**	65.25	43.10	53.05
Precision	**92.5**	**83.49**	**74.49**	44.68	4.27	17.1
Recall	**88.68**	**75.58**	**69.86**	28.68	8.45	15.2
F1-score	**89.92**	**78.62**	**71.10**	28.00	5.63	11.35
Balance Accuracy	**88.67**	**75.57**	**69.85**	23.31	9.49	17.08
Weighted Precision	**96.73**	**91.05**	**83.17**	77.05	21.24	36.69
Weighted Recall	**96.61**	**91.48**	**84.29**	71.97	43.99	52.46
Weighted F1-score	**96.55**	**91.09**	**84.43**	67.93	28.45	39.40

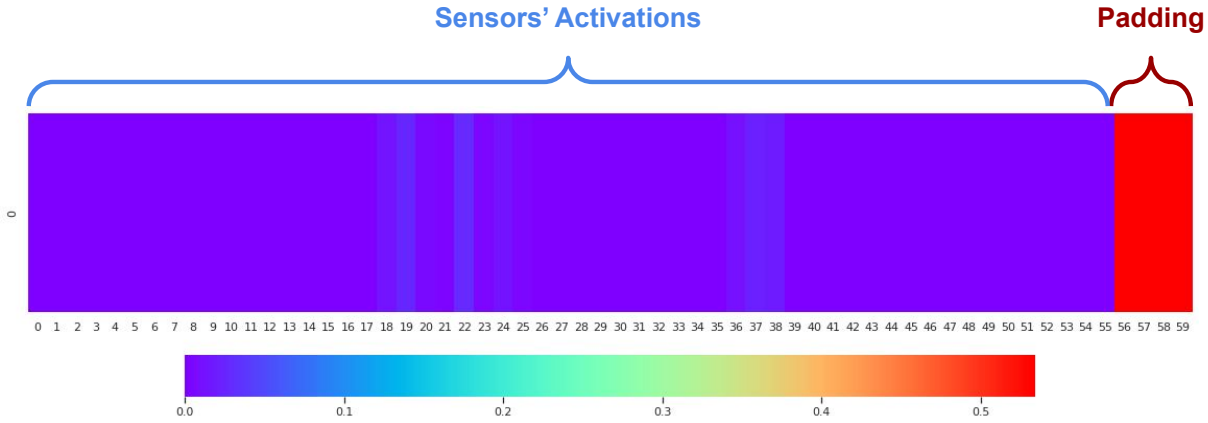
The results presented in the Table 4.3 summarize scores obtained after training the algorithms as described above. By looking at this table, we can see that the FCN have a very strong degradation of its performance compared to the LSTM in the previous experiment. This drop in performance could be explained by the zero-padding used to obtain sequence of same length. Where the LSTMs is able to ignore the padding, the FCN might interprets it as useful data. Other works have shown that the processing of time series of different lengths is problematic [17] and methods to re-scale or re-sample could be applied. However, in our case of time series in

the form of events, re-sampling is not possible because the length of the sequences of activities is information that may be important.

By using the Gradient-weighted Class Activation Map (Grad-CAM) method proposed by Selvaraju et al. [14] we can visualize what the FCN focuses on to determine the label of the sequence. Grad-CAM is used for the deciphering of convolutional neural networks and consists, in the context of an image, of searching for which parts of the image led the network to its final decision. It produces heatmaps representing the activation classes on the images received as input. An activation class is associated with a specific output class. These classes will indicate the importance of each pixel in relation to the class concerned by increasing or decreasing the intensity of the pixel. Here, we have applied this method to the FCN for a sequence problem.



(a) Zero padding visualization: full sequence



(b) Zero padding visualization: zoom on the beginning of the sequence

Figure 4.5 – Exemple of a Gradient-weighted Class Activation Map (Grad-CAM) of an activity sequence. Red means strong focus.

Figure 4.5 represents the Grad-CAM of an activity sequence and a zoom in on the beginning of the sequence with the area containing the sensor activations. Each colored column corresponds

to a sensor activation. As can be seen in Figure 4.5a, the FCN places great importance on padding, which distorts the result. Indeed, the padding should not be part of the pattern being recognized. We can see through the zoom in on the beginning of the sequence in Figure 4.5b, that the padding seems to be more important than the useful data located at the beginning of the sequence. The FCN therefore learns where the padding is located and the size of the padding, but does not extract a useful pattern. This interpretation of the padding by the FCN is caused by the representation of the padding generated by the embedding. In fact, the embedding assigns a representation vector to this value 0.

4.5.3 Padding Methods Comparison

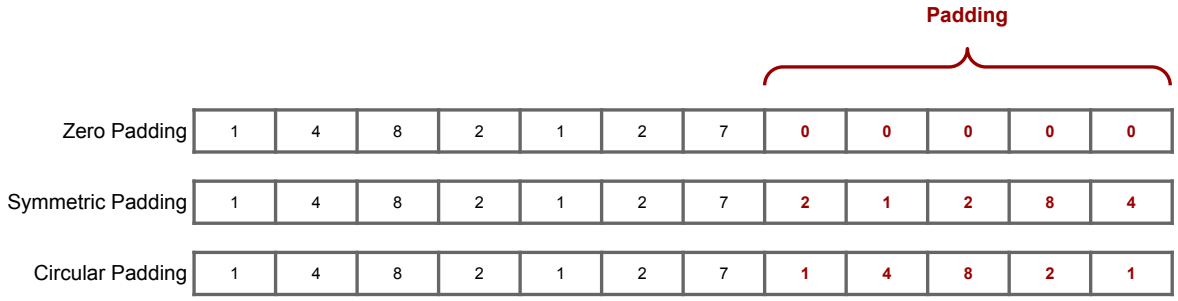
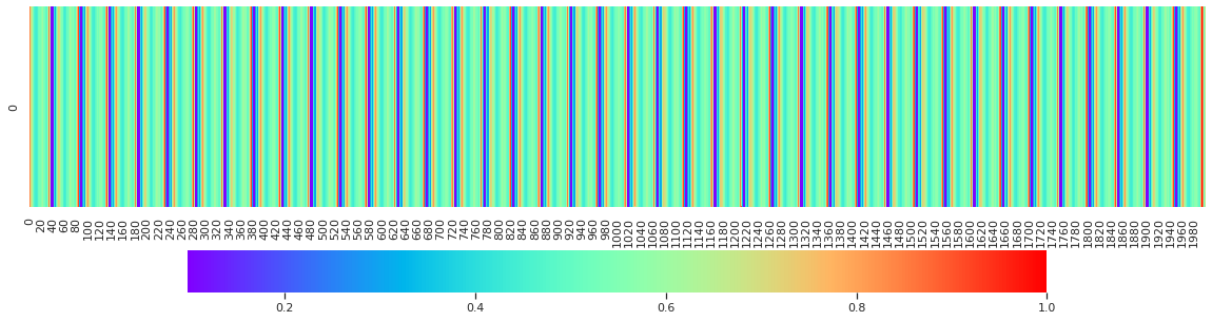


Figure 4.6 – Example of zero padding, symmetric padding and circular padding

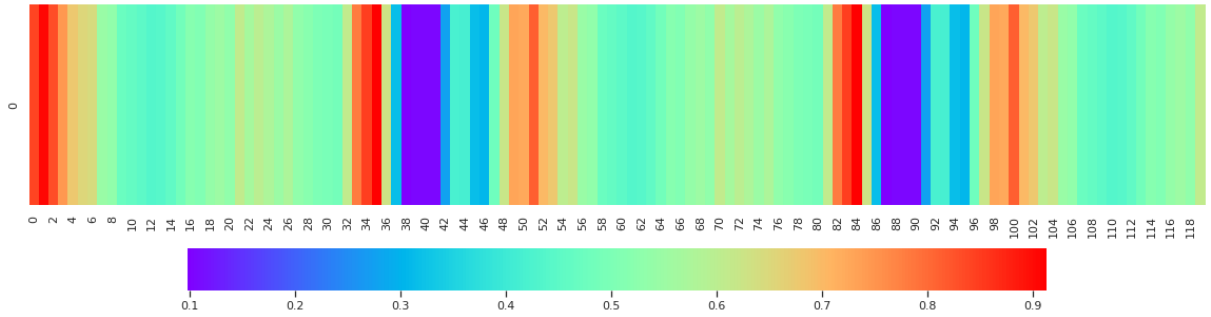
To attempt to solve the zero-padding problem, we replaced this type of padding with others, as illustrated in Figure 4.6: specifically symmetric-padding [20] and circular-padding [13]. Instead of adding to the end of the sequence of zeros, the symmetric-padding instead repeats the sensors' activations according to an axis of symmetry. The circular padding, on the other hand, repeats the complete value sequence infinitely. These types of padding act as data augmentation and force the FCN to focus on patterns.

By using a Grad-CAM on a sequence completed with a circular padding for example, see Figure 4.7a, we can see the repetition of the pattern and thus the sensor activations repetition. If we zoom in as we did previously on the beginning of the sequence, see Figure 4.7, we can clearly see the pattern the FCN is focused on. As we can observe, the FCN is always focused on the same pieces of sensors that seems to be the beginning and the end of the sequence. Some activations in the middle seem also important for this activity.

As visible in Table 4.4 the symmetric padding allows FCN to obtain the highest scores on the Aruba and Milan datasets. However, it degrades the performance of the FCN over the Cairo dataset. Meanwhile, we can see that the scores are more balanced with the use of circular padding, with the only main concern being that this technique does not significantly improve results for the Cairo dataset. We can potentially explain the poor scores on the latter by the



(a) Circular padding visualization: full sequence



(b) Circular padding visualization: zoom on the beginning of the sequence

Figure 4.7 – Exemple of a Gradient-weighted Class Activation Map (Grad-CAM) of an activity sequence with a circular padding. Red means strong focus.

Table 4.4 – Comparison between zero padding, symmetric padding and circular padding

	FCN zero-padding			FCN symmetric-padding			FCN circular-padding		
	Aruba	Milan	Cairo	Aruba	Milan	Cairo	Aruba	Milan	Cairo
Accuracy	65.25	43.10	53.05	**95.01**	**82.18**	36.46	94.09	80.12	**53.3**
Precision	44.68	4.27	17.1	**78.94**	**69.87**	15.61	77.86	65.07	**21.29**
Recall	28.68	8.45	15.2	**77.54**	**66.43**	**16.16**	73.73	66.48	14.19
F1-score	28.0	5.63	11.35	**77.41**	**65.42**	10.5	73.84	64.62	**13.67**
Balance Accuracy	23.31	9.49	17.08	**76.79**	**66.36**	**19.57**	72.47	65.11	17.15
Weighted Precision	77.05	21.24	36.69	**94.46**	**81.85**	**42.95**	93.61	80.35	35.01
Weighted Recall	71.97	43.99	**52.46**	**94.85**	**82.15**	43.26	93.93	79.87	51.83
Weighted F1-score	67.93	28.45	**39.4**	**94.55**	**81.42**	37.35	93.58	79.41	37.56

fact that it is a dataset containing two residents and is therefore more complex. The patterns are less easily identifiable because the sequences of one resident are affected by the actions of the second resident. Padding is certainly a problem for FCN but it seems that pattern recognition is not enough to recognize activities.

4.5.4 FCN and LSTM Final Comparison

The strength of FCN is its ability to recognize patterns while LSTM uses temporal relationships in addition to patterns. Comparing the best results, or at least the most balanced ones, obtained with the circular padding against those obtained with the bidirectional LSTM (see Table 4.5), we can see that the scores of the bidirectional LSTM are largely superior. In particular, the scores of the bidirectional LSTM are most effective for not only the Cairo dataset, but also the Milan. We can clearly see that pattern recognition alone is not sufficient for full recognition of activities. Rather, it is necessary to interpret the temporal relationships within the sequence. In comparison, despite FCN being a structure known for its ability to extract relevant features and its performance in terms of time series classification, is not sufficient for activity sequence classification.

Table 4.5 – Comparison between embedding + bidirectional LSTM and embedding + FCN with circular padding

	embedding + bidirectional LSTM			embedding + FCN		
	Aruba	Milan	Cairo	Aruba	Milan	Cairo
Accuracy	**96.60**	**91.47**	**85.32**	94.09	80.12	53.3
Precision	**92.5**	**83.49**	**74.49**	77.86	65.07	21.29
Recall	**88.68**	**75.58**	**69.86**	73.73	66.48	14.19
F1-score	**89.92**	**78.62**	**71.10**	73.84	64.62	13.67
Balance Accuracy	**88.67**	**75.57**	**69.85**	72.47	65.11	17.15
Weighted Precision	**96.73**	**91.05**	**83.17**	93.61	80.35	35.01
Weighted Recall	**96.61**	**91.48**	**84.29**	93.93	79.87	51.83
Weighted F1-score	**96.55**	**91.09**	**84.43**	93.58	79.41	37.56

4.6 Summary

This chapter presented a hybrid approach that combines an input representation from the NLP domain and a classifier algorithm from the TSC domain to create an end-to-end model. Our assessment on two datasets with SEW over activity sequences shows that the embedding significantly improves the performance of LSTM and FCN in all cases. This means that the domain knowledge incorporated in the embedding can improve the understanding of the sensors' activation by LSTM and FCN.

By using SEWs we saw that the FCN obtained the same or greater performance than the LSTM, with the exception of two configurations only. We also found that by in large the FCN was quicker to train. Moreover, the FCN outperformed the LSTM when the window size decreased.

This seems means that the FCN can extract meaningful patterns.

Nevertheless, when the FCN is compared to the LSTM over full activity sequences with zero-padding, the LSTM outperforms drastically. We tried to solve this padding problem by using two other padding methods, circular-padding and symmetric-padding. The circular-padding method improved the FCN results to hell the LSTM ones. This padding approach showed the highest performance and most balanced results on the three studied CASAS datasets. Further, the method demonstrated that pattern recognition can be used to get high ADLs recognition scores over two datasets Aruba and Milan. Unfortunately, the FCN approach combined with the circular padding did not outperform the bidirectional LSTM approach. This gap is even more visible on the Cairo dataset, where the FCN approach is still very far behind the LSTM method.

In the first experiment as all SEW were shuffled and split into train and test sets, the data of subsets was similar, thus only the pattern recognition was necessary. However, the second experiment focuses on the ability of models to extract the information over all the activity sequences. In addition, in this second experiment the data of the train and test subsets were less similar.

We saw with this last experiment, that temporal dependencies were also important, which allowed the LSTM to outperform the FCN. Indeed, patterns in datasets with pets, such as Milan, or with multiple residents, such as Cairo, are less easy to extract, due to the potential noisy sequences. These datasets need more than just pattern recognition. They need sequential and temporal relations understanding, to extract the actually important information, because some sensor activations are temporally correlated and linked with the current activity while others are not. In the next chapter we will explore how to improve the embedding layer to encode, understand, and tackle the question of long temporal dependencies and sensor's activation context.

BIBLIOGRAPHY

[1] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, 2012.

[2] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1367–1376. IEEE, 2018.

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[4] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[7] Daniele Liciotti, Michele Bernardini, Luca Romeo, and Emanuele Frontoni. A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 04 2019. doi: 10.1016/j.neucom.2018.10.104.

[8] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[10] Tin Lay Nwe, Tran Huy Dat, and Bin Ma. Convolutional neural network with multi-task learning scheme for acoustic scene classification. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1347–1350. IEEE, 2017.

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

[12] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018.

[13] Stijn Schoeters, Wim Dewulf, Jean-Pierre Kruth, Han Haitjema, and Bart Boeckmans. Description and validation of a circular padding method for linear roughness measurements of short data lengths. *MethodsX*, 7:101122, 2020.

[14] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[15] Gian Antonio Susto, Angelo Cenedese, and Matteo Terzi. Time-series classification methods: Review and applications to power systems data. *Big data application in power systems*, pages 179–220, 2018.

[16] Syed Fahad Tahir, Labiba Gillani Fahad, and Kashif Kifayat. Key feature identification for recognition of activities performed by a smart-home resident. *Journal of Ambient Intelligence and Humanized Computing*, 11(5):2105–2115, 2020.

[17] Chang Wei Tan, François Petitjean, Eamonn Keogh, and Geoffrey I Webb. Time series classification for varying length series. *arXiv preprint arXiv:1910.04341*, 2019.

[18] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

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

[20] Shuang Wu, Guanrui Wang, Pei Tang, Feng Chen, and Luping Shi. Convolution with even-sized kernels and symmetric padding. *arXiv preprint arXiv:1903.08385*, 2019.

[21] Surong Yan, Kwei-Jay Lin, Xiaolin Zheng, and Wenyu Zhang. Using latent knowledge to improve real-time activity recognition for smart iot. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[22] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

Chapter 5

Language Modeling for Human Activity Recognition

5.1 Introduction

The common points of previous approaches using deep learning algorithms for the human activity recognition in smart homes is that they extract automatically features and patterns from raw data. LSTM-based approaches have demonstrates the best results as they can capture patterns and temporal dependencies. However, they ignore some long dependencies, the semantics, the sensor type, the syntax and the context in which the sensor is activated.

Semantic analysis, context modeling and words relations have been, very early and naturally, at the center of NLP research. The latest advances in this field have proposed various unsupervised pre-training methods of projecting words into a vector space, called embedding, to create word representations and language models [5, 7]. These embeddings capture information about the construction of words, sentences and texts. They are also able to capture the context of a word in a document, the semantic or the syntactic similarities, relations with other words, etc. They have proven to be effective in transfer learning on translation, text generation and classification tasks. Embedding in the NLP field can be classified into two categories. Static embedding approaches, such as Word2vec [11], GloVe [12], FastText [2], etc. And contextualized approaches, such as ELMo [13], BERT [4], GPT [14], etc.

In the context of human activity recognition, previous work has already investigated the use of embedding methods. The training of a Word2Vec model was used to cluster and create a semantic relations map of habits of a population in a city [3]. The use of a public pre-trained, on texts, Word2Vec model was used to associate a label with unknown activities in the context of worn sensors [9, 16]. The same approach has been studied to annotate unknown activities and perform "zero-shot" learning [6] in a smart home [1].

In the previous chapter we saw that an embedding layer improve models classification results. Unsupervised embedding training, as used in the NLP domain, but for sensor activations, has not been exploited. Capturing the context and semantics of sensor activation is not exploited either. However, thanks to the NLP embeddings' generation methods, it would be possible to capture

this information of sensor activations. This type of information would enrich the knowledge on the activation in question and would allow distinguishing it according to the cases of appearance.

In previous chapters, we have already drawn attention to the fact that home automation sensors or sensors embedded in everyday objects provide little information. It is the type of sensor, the type of activation, the order and the context composed of sensor activations before and after a given sensor, that provide richer information that can be better exploited. This why bidirectional LSTM approaches seems to better perform than unidirectional LSTM ones, in addition to the ability of bidirectional LSTM to better encode long sequences.

This last remark already indicates that it would be necessary to study some notion of semantics, syntax and context related to each activation. Indeed, the encoding of the ith activation data should not depend only on its value and be expressed as a function $f(activation_i)$. Instead, it should capture more information on sensor behavior, incorporating relevant neighboring activations while smoothing out variability and noise. In other words, the sensor encoding at ith activation should also depend on the other activations in the sequence, and be expressed as a function $f(activation_i, \{activation_1, activation_{i-1}\}$,
$\{activation_{i+1}, activation_n\})$, where n is the number of activations in the sequence. This could improve the performance of ADL classification algorithms. Moreover, the creation of a pre-trained embedding could allow, by transfer learning, to perform the human activity recognition task in new houses.

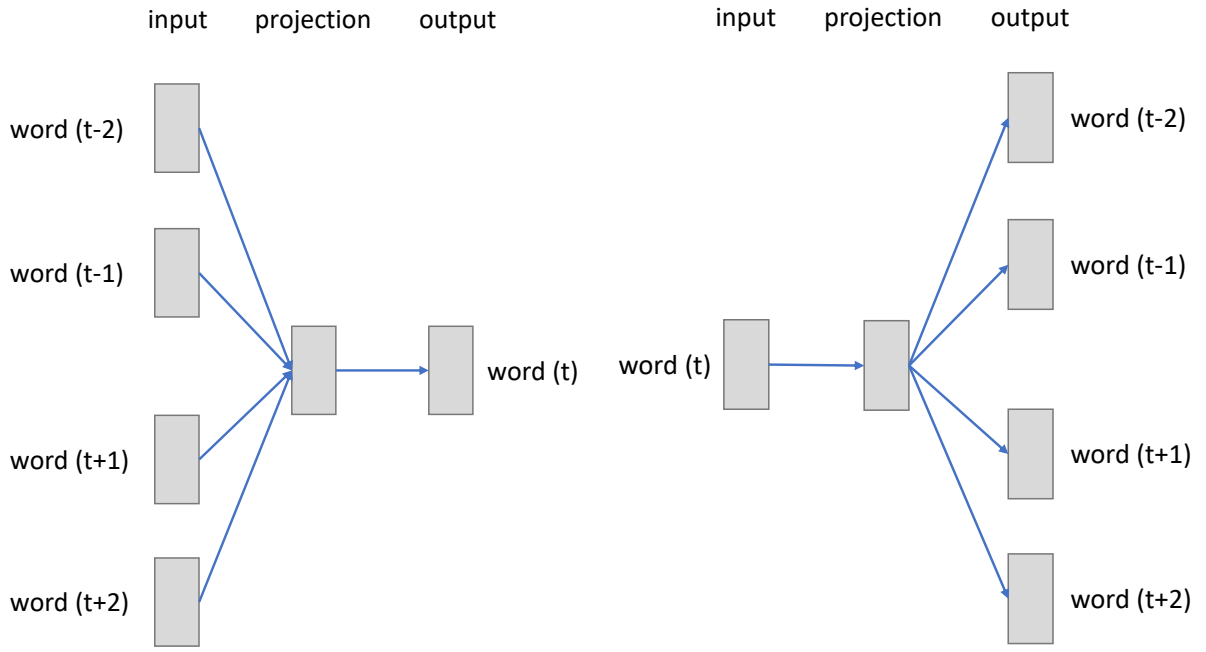
This chapter focuses on using advanced embedding methods inspired by NLP domain to obtain a better sensor representation in order to improve a LSTM-based classifier. First, we will dive into the world of semantic embedding with Word2Vec and see what kind of information and functionality such a model can provide. Second, we will explore the language model principle and contextualized embedding domain with ELMo. Third, we will evaluate and observe the potential improvement of using advanced pre-train embeddings. Finally, we are going to evolve if it is possible to transfer knowledge from one house to another via an embedding.

5.2 Sensor Events: The Semantic and Context Embedding

5.2.1 Word Embedding : Word2Vec

Word2Vec is one of the most popular techniques to learn word embeddings using shallow neural networks. Proposed by Mikolov et al. [11], Word2Vec is an unsupervised learning technique to learn continuous representations of words. In many ways, Word2Vec builds on a bag of words, but, instead of assigning discrete values to words, it learns a continuous multi-dimensional vector representation for each word in the training corpus. There exist two training modes for learning representations of words with the Word2Vec algorithm, the Continuous Bag Of Words (CBOW) and the Skip-Gram modes.

As seen in Figure 5.1, these two methods are essentially the reverse of each other. In both methods, all the words in the vocabulary are one-hot encoded and a window size is defined to consider a fixed number of words. The CBOW is trained by learning to predict the center word in each window based on its surrounding words (context words), as in Figure 5.1a. Once a prediction is made, the difference between the predicted vector and the ground truth is backpropagated to update the dense layer that define the embedding. On the other hand, the Skip-Gram method is trained to predict, given a target word, the most probable words in a fixed sized window around it (context words), as in Figure 5.1b. Generally, Skip-gram is better with a larger corpus but is slower to train, while CBOW is more suitable if the corpus is small or speed is a factor.



(a) Continuous Bag Of Words

(b) Skip-Gram

Figure 5.1 – Word2Vec methods.

By design, Word2Vec captures the similarity of words in a corpus. Moreover, thanks to the distance of a word from other words it calculates, Word2Vec also captures part of the meaning of the word. CBOW, for example, learn better syntactic relations between words while Skip-gram is better in capturing better semantic relations. In practice, this means that for the word “cat”, CBOW would retrive as closest vectors morphologically similar words like plurals, i.e. “cats”. On the other hand, Skip-gram would consider morphologically different words (but semantically relevant) like “dog” much closer to “cat” in comparison.

Moreover, because Skip-gram rely on single words input, it is less sensitive to overfit frequent words. Indeed even if frequent words are presented more times that rare words during training,

they still appear individually. CBOW is prone to overfit frequent words because they appear several time along with the same context. This advantage over frequent words overfitting leads Skip-gram to be also more efficient in term of documents required to achieve good performances, much less than CBOW. It's also the reason of the better performances of Skip-gram in capturing semantical relations.

5.2.2 Word2Vec for Sensor Event Embedding

As it is made for words, we have trained a Word2Vec model for sensors in order to capture sensors and sensors' activation relations. To do so, we used the Skip-Gram method, described above, to predict surround sensor activations of a middle one in a window. This choice was motivated by the strong ability of the Skip-Gram methods to be more semantically relevant. Indeed, we don't want to learn only if activations from a same sensor are close or not, but we also expect to capture relations between sensors' activation.

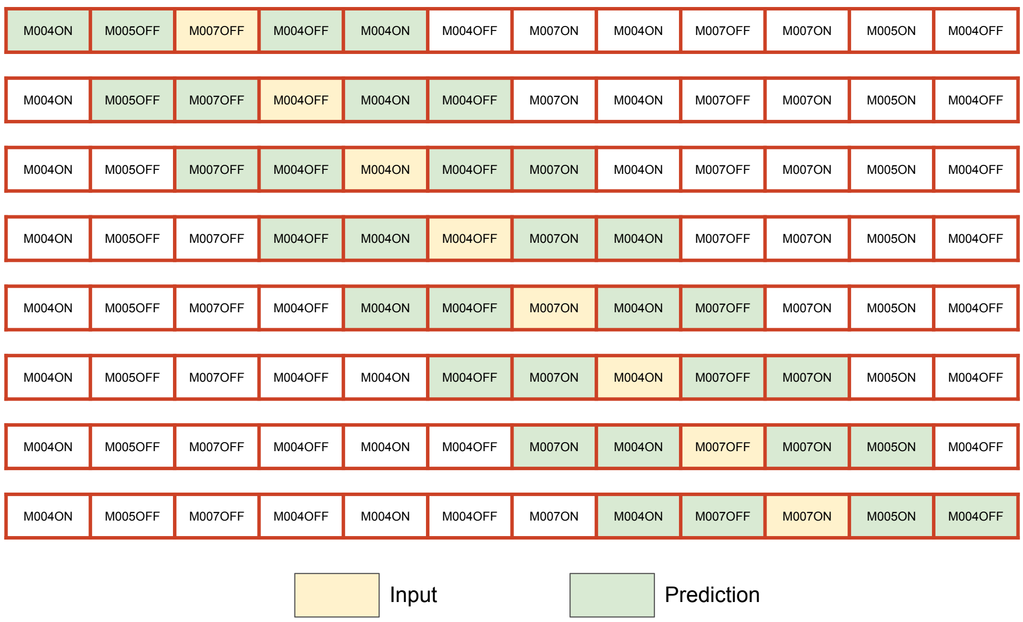


Figure 5.2 – Windowing example with a window of size 5

Windows used for the Skip-Gram method, are subsets of sensors' activations from activities sequences. In other words, each activity sequence is split into subsets of sensors' activations via a sliding window with a stride of one, see example in Figure 5.2. All windows are then used for training, using the central activation sensor as input and the surrounding activations as output (prediction), see Figure 5.3. The result of the training is the generation of sensors' activation embedding that we can call **"Sensor2Vec"**. To train our Sensor2Vec model we used the hyperparameters in Table 5.1

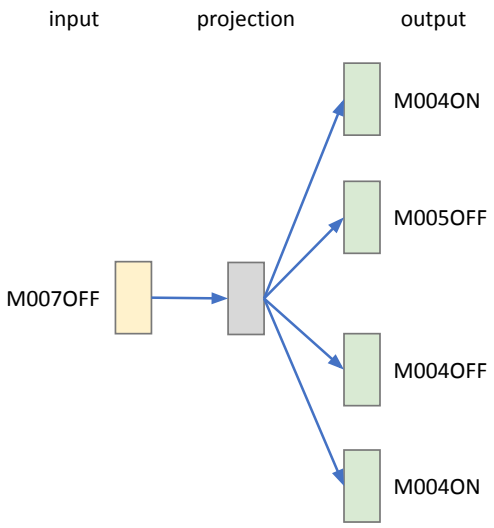


Figure 5.3 – Sensor activations Skip-Gram example

Table 5.1 – Word2Vec embeddings hyperparameters

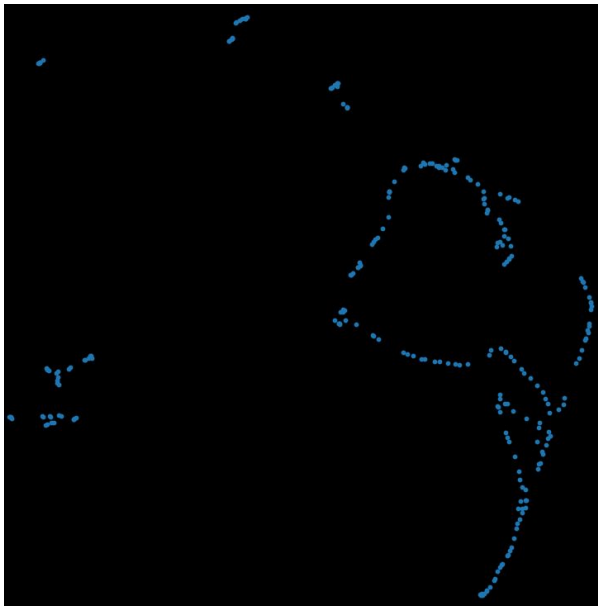
Embedding size	64
Context windows size	60
Max epochs number	100

5.2.3 Sensor2Vec: Sensors' Activations Visualization

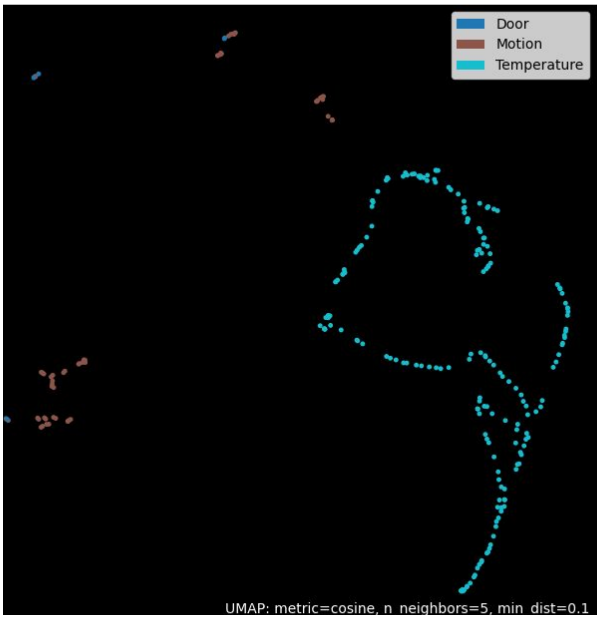
Once the embedding of the sensors is trained, it is possible to interpret the features generated by the model by visualizing the embedding vector of each sensor activation. To do this, each embedding vector is projected into a 2D or 3D space. Therefore, as embedding vectors have a d-dimensions size, they should be reduced. After extracting the vector representation of each sensor activation, the Uniform Manifold Approximation and Projection (UMAP) algorithm [10] is used to reduce vectors' dimension.

UMAP is a recent dimension reduction technique that can be used for visualisation, but also for general non-linear dimension reduction. The algorithm is founded on three assumptions about the data: 1) The data is uniformly distributed on Riemannian manifold; 2) The Riemannian metric is locally constant (or can be approximated as such); 3) The manifold is locally connected.

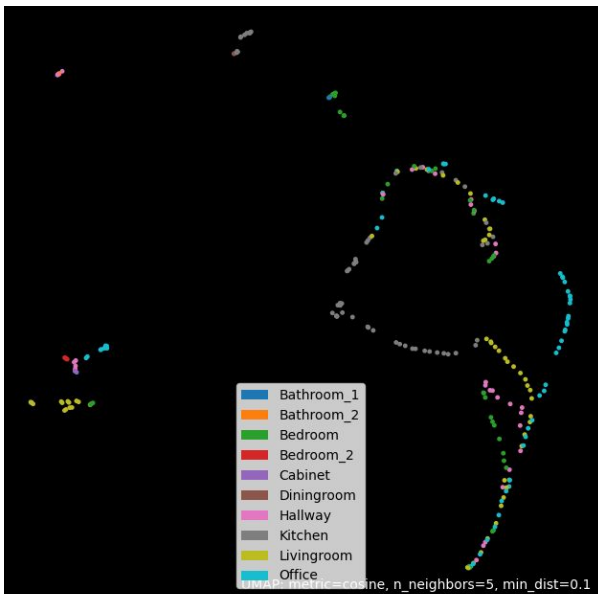
Once reduced to 2D vectors, each of them can be plotted. For example, the Figure 5.4a is the sensors' activation projection of vectors from Aruba dataset. At a first look, several clusters can be seen. Coloring each point with a color that corresponds to the type of sensor, as in Figure 5.4b, outlines that temperature sensors (light blue) are clustered all together, and other clusters seems to be composed mainly of motion sensors. The model has learnt that temperature sensors have not particular links with others sensors. It seems that the model learned the sensor nature.



(a) raw clusters



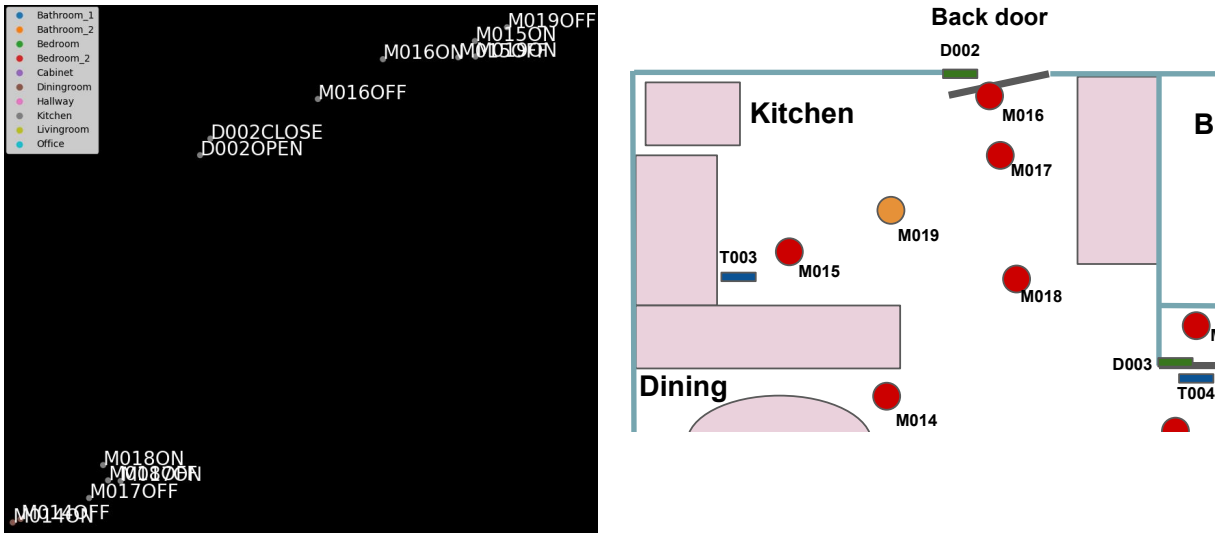
(b) colorized by sensors' nature



(c) colorized by room

Figure 5.4 – Aruba sensors' activation embedding visualization

However, we can observe that some clusters contain door sensors, which means that the clusters must represent another type of information. By changing the color of each of the points by their location, as in Figure 5.4c, it appears that, in general, each cluster corresponds to a room of the Aruba house.



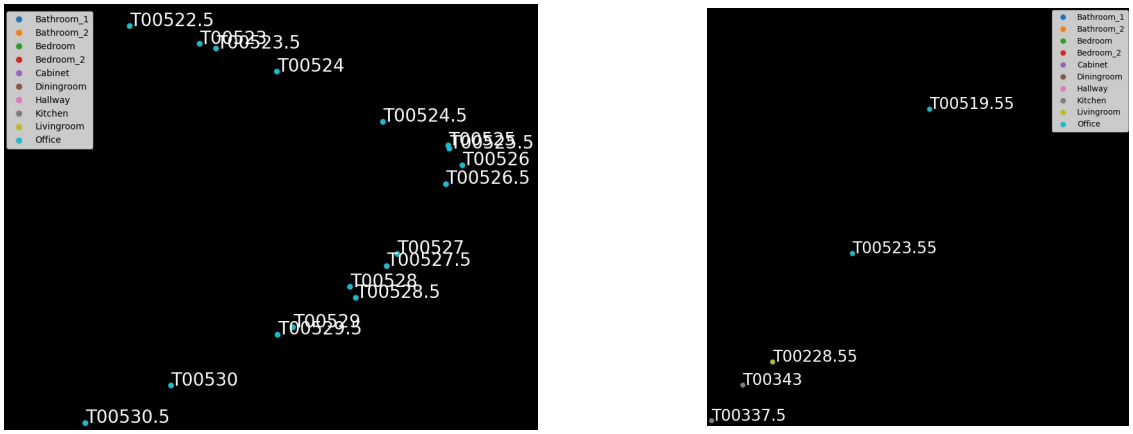
(a) Zoom in Kitchen cluster in top

(b) Aruba kitchen house map

Figure 5.5 – Room cluster

A zoom on the grey cluster, “kitchen”, on the top of the room colored picture, is provided in Figure 5.5a and can be compared to the floor map of the house in Figure 5.5b. All the sensors associated with the kitchen on the plan are in the zoomed cluster. Two sensors’ activation, “M014ON” and “M014OFF”, that aren’t in the “kitchen” but belong to the “dining room” appear close to some kitchen sensors (M018 and M017 activations). This close distance is quite coherent as they are physically close and are correlated to zones of passage between these two rooms. After examining the different clusters, they all seem to correspond to a room in the house. This grouping into rooms is quite consistent because ADLs are usually located in one room of the house. It is important to note that, in an unsupervised way, the Word2Vec method has captured the notion of sensor localization by grouping the sensors belonging to the same room. This sensor localization clustering was also observed by Singla and Bose [17] in their work on IoT devices identification with binary sensors.

Concerning the activation of the temperature sensors, it seems that they are also grouped in sub clusters belonging to the same room. If we zoom in the main office temperature sensor cluster, we can observe that temperature are ordered according to temperature values, see Figure 5.6a.



(a) Cluster of Aruba office Temperature sensor

(b) High kitchen temperature correlations

Figure 5.6 – Room cluster

Moreover, some temperatures belonging to different rooms are close to each other. The model seems to find correlations between some temperatures coming from different rooms. For example, if we look at the frontier of high kitchen's temperatures, we can observe that when the kitchen temperature is very high, superior to 35 degrees, the living room temperature is under 30 degrees and the office temperature sensor is between 20 and 24 degrees.

Generally, in the Aruba dataset a high temperature in the kitchen means that the resident is cooking. This sensor correlation between the kitchen, the office and the living room temperature sensors should help the classifier to better understand when the resident is cooking.

Here, we have illustrate only the case of the Aruba dataset. However, we can observed similar results on all the evaluated datasets, see Appendix B.

5.2.4 Sensor2Vec: Activity Sequences Embedding Visualization

In the previous section, we have visualized and attempted to interpret the features learned by the Sensor2Vec model. We will now plot the embedding of each activity sequence in order to see if some activity clusters appear thanks to the features provided by the Sensor2Vec model. Aiming at visualizing this embedding of activities sequences, we created a model composed of the Sensor2Vec layer and a GAP layer [8] (see Figure 5.7).

The Sensor2Vec layer provides a feature vector representation for each sensor activation in the input sequence. The GAP layer summarizes these feature vectors into one vector which is the activity sequence embedded vector. Once all activation sequences are turned into a vector, as for Sensor2Vec visualization we use the UMAP to reduce each activity vector to a two-dimensional vector. Each activity vector is then displayed and tagged with a color corresponding to the activity label.

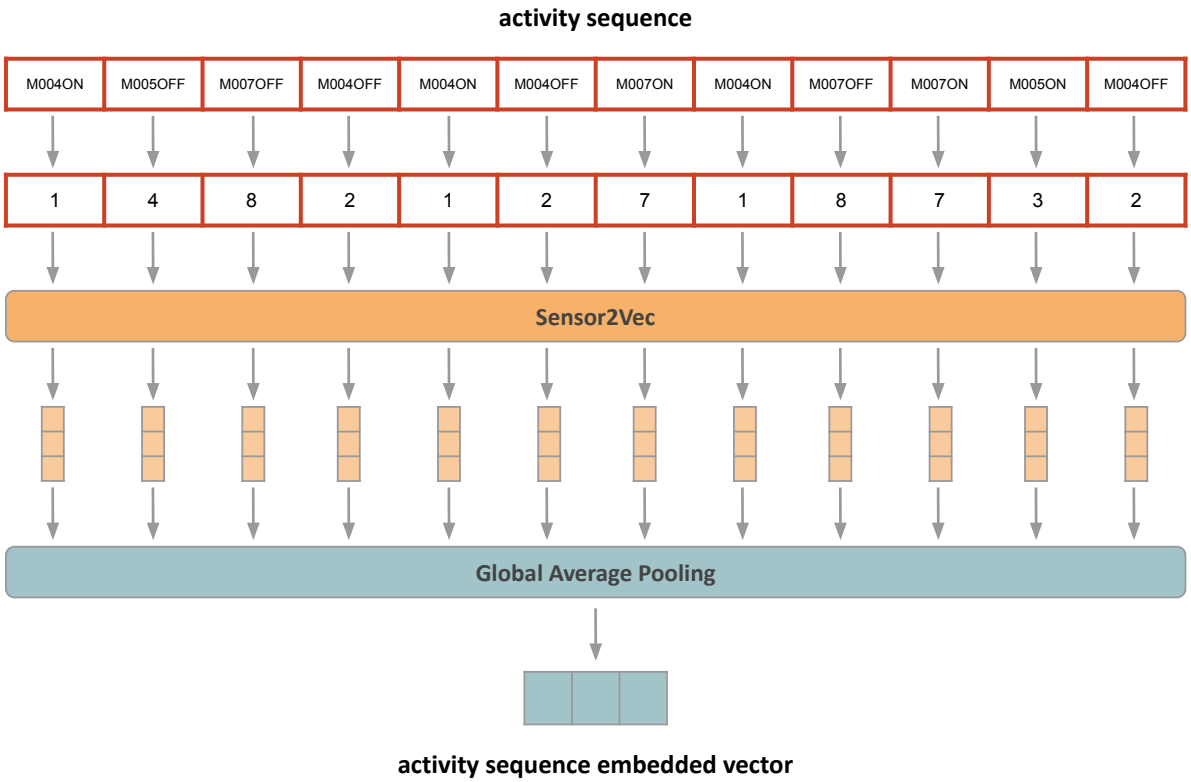


Figure 5.7 – Sensor2Vec activity sequence embedding model

The Figures 5.8, 5.9 and 5.10 show the activity sequence embedding of the Aruba, Milan and Cairo datasets respectively. Each dot on figures is an activity sequence. To provide the embedding of activity sequences, we use the Sensor2Vec embedding corresponding to the right dataset. In other words, we pay attention to use the Aruba Sensor2Vec model to embed Aruba's activity sequences, and do the same for the other datasets.

At a first look, this representation allows us to assert that a Sensor2Vec model is able to extract features that can classify the activation sequences. We can observe on the three datasets that activities with the same label are almost grouped together. The activities corresponding to the "Other" class are mainly concentrated in the center of the representations, while the other activity classes are found at the periphery and very few distinct clusters appear.

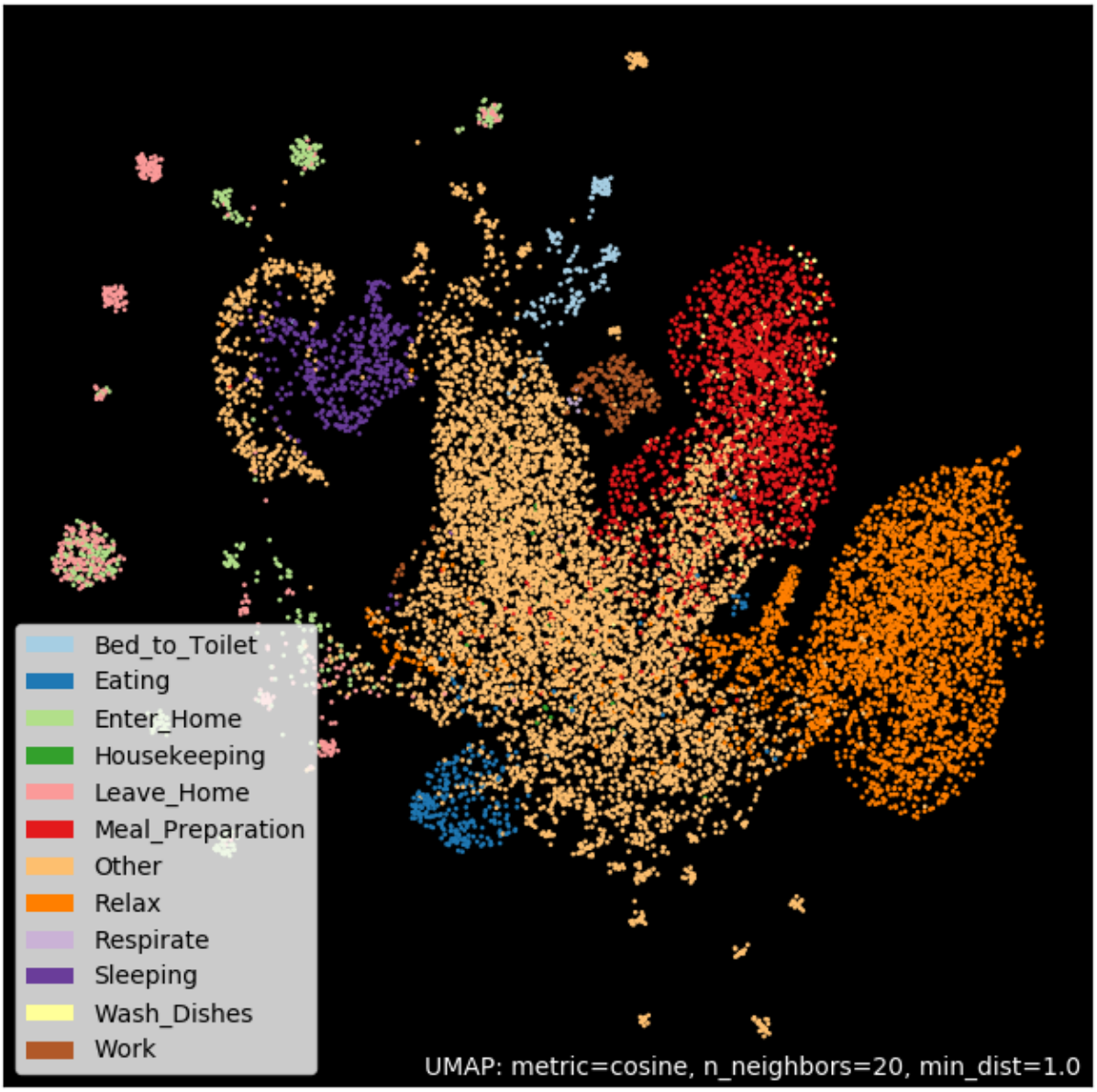


Figure 5.8 – Activity sequences of Aruba embedded by Word2Vec

On the Aruba dataset figure, we can see the "Sleeping" activity cluster that is almost isolated, as the "Bed to Toilet" and "Work" activities clusters. It seems that an appropriate classifier should easily find boundaries between these clusters. However, this embedding method does not seem to be efficient enough to isolate in individual clusters all the activity classes. The majority of the clusters are connected by the points corresponding to the activity sequences labeled "Other". Similar activities such as "Enter Home" and "Leave Home" are still mixed. The more complex the dataset, the less the activities are grouped into clusters.

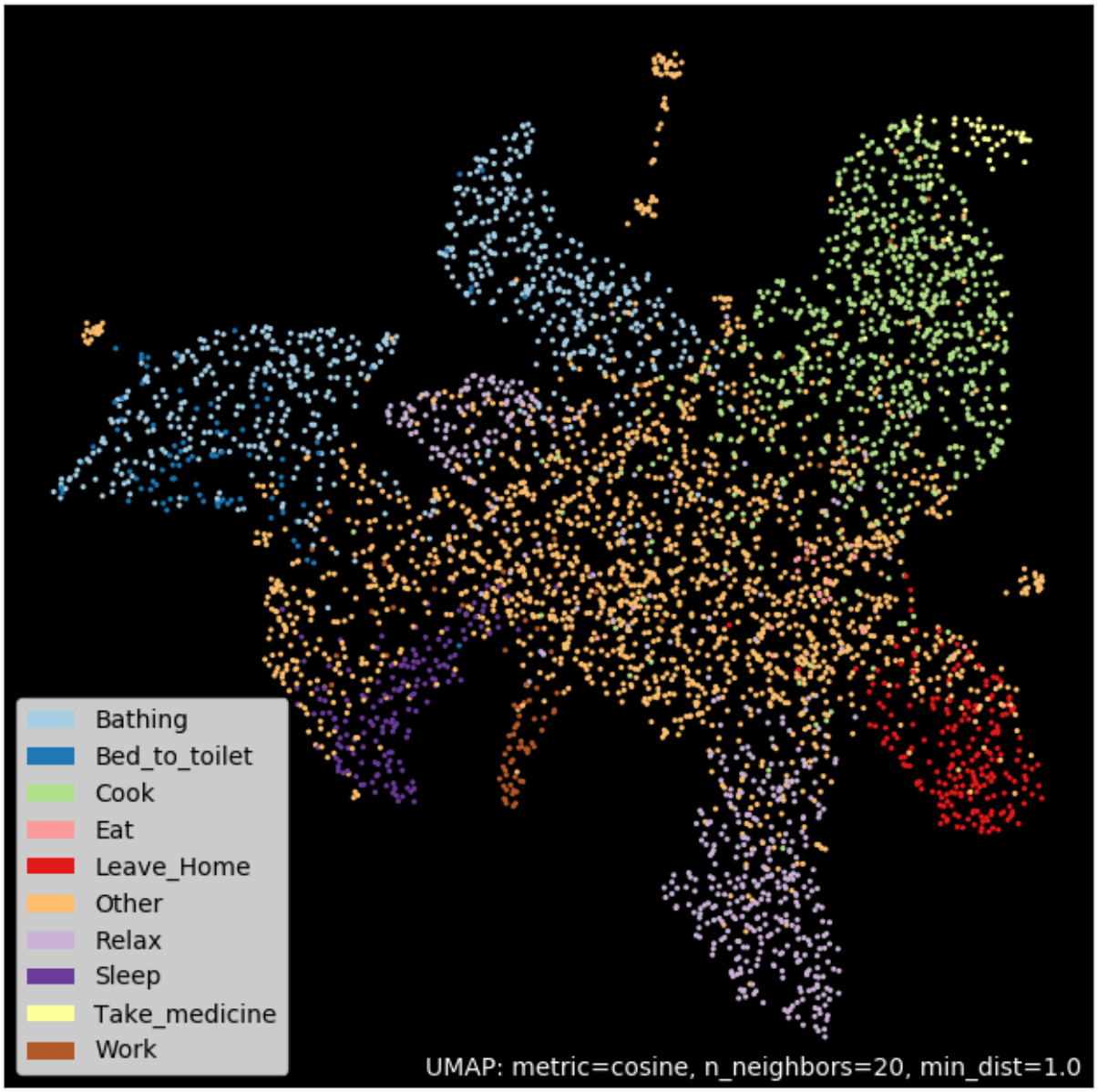


Figure 5.9 – Activity sequences of Milan embedded by Word2Vec

In the Milan figure, activity clusters are in the periphery but not completely isolated. The "Bed to Toilet" activity is mainly mixed with the "Bathing" activity. Two clusters corresponding to the activity "Bathing" appear, however it seems possible as the Milan house has two bathrooms. Only the "Work" activity cluster looks mainly isolated. This lets us think that this activity will be well recognized compared to the others. Generally for this dataset activities seem mainly recognizable. However, it seems that activities will have a high confusion with the "Other" class.

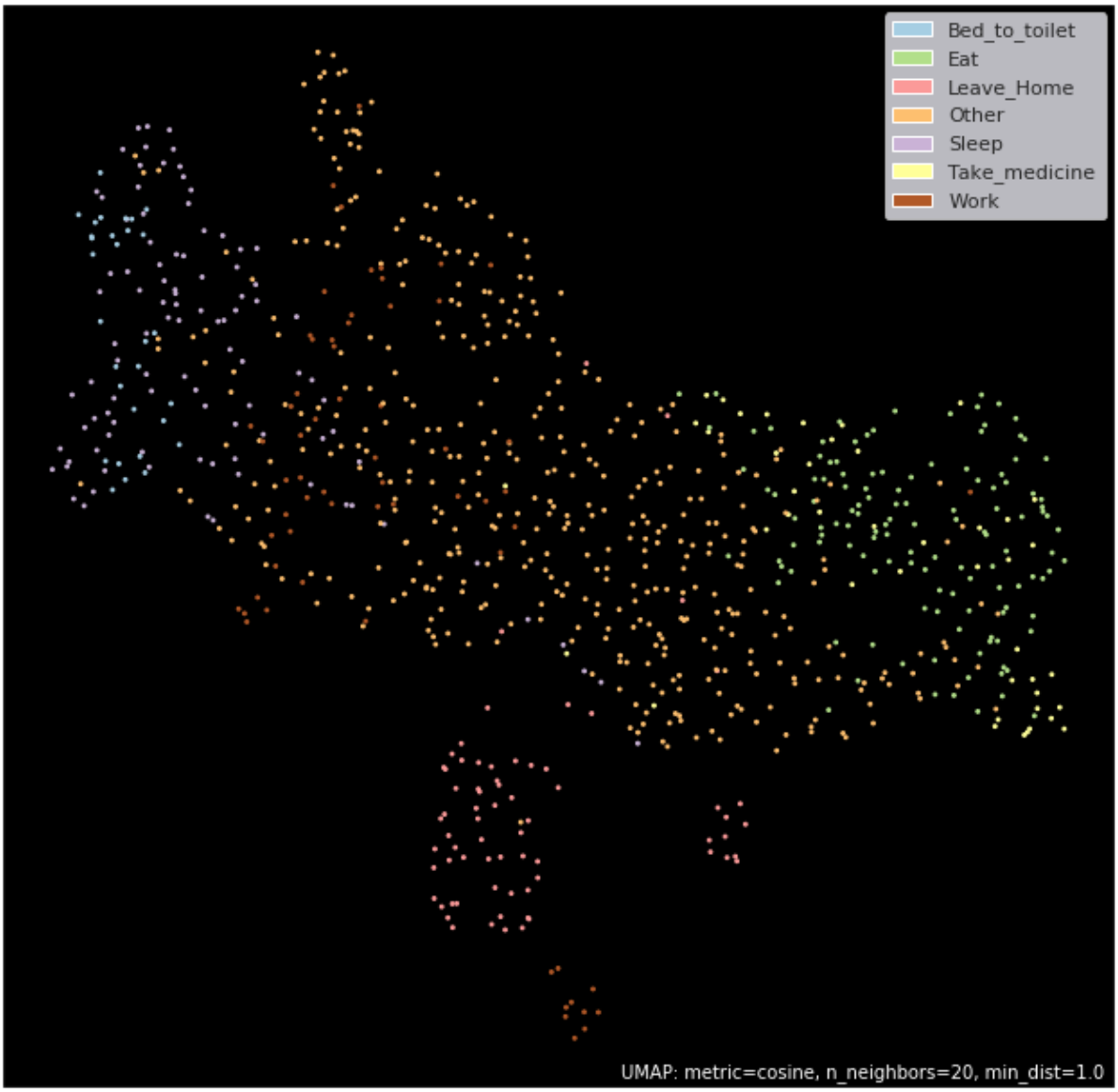


Figure 5.10 – Activity sequences of Cairo embedded by Word2Vec

For the Cairo dataset the both activities "Sleep" and "Work" are in majority isolated. However, all "Bed to toilet" dots are mixed with a subset of "Sleep" dots. The activity "Take Medicine" is completely absorbed in the "Eat" dots activities. We will expect the baddest classification results on this dataset with this embedding method.

This pre-trained embedding method seems promising to improve a ADLs recognition algorithm. Coupled with the UMAP algorithm, in an unsupervised way, let's generate almost isolated activity clusters. However, even if the Word2Vec method is a powerful embedding technique and allows us to create a meaningful Sensor2Vec embedding, this embedding method has a main drawback. It provides a single representation for each word / sensor, regardless of the context. Put differently in the NLP field, the word "orange" in syntagms, such as "the orange juice" or "the orange car", garners the same vector representation, although "orange" does not have the same meaning in the two sequences of words. This lack of context understanding is an important issue to capture the sentence's meaning and introduce the well-known polysemy problem.

As said earlier, sensors meaning information depends on surrounding sensors' activation such as words in the sentence, cf. above example. The NLP field provides advanced embedding methods that can understand word context, and can represent the same word according to its context. In the next section, we will study one of them for the ADLs recognition problem.

5.3 Sensor Events: The Syntax and Context Embedding

5.3.1 The Language Model Principe

Before diving into the world of advanced embedding methods, it is important to understand the language model technique used in the NLP field. To illustrate what is the intuition behind a language model, let's take the following sentence example, "Project A has an elevated risk of ...". By reading this incomplete sentence, we are able to predict the next word after "of". Between the two words "delay" and "water" the most probable word seems "delay". Indeed, we have some knowledge about how a sentence is built according to rules and the sentence context. This knowledge is called a language model. More formally, the NLP field define a language model, basically as a probability distribution over words or word sequences. Thus, to generate language models, NLP researchers train deep learning models on the task, which is simply to assign a probability to any sequence of words (see Eq 5.1).

$$P(w_i|w_1, w_2, ..., w_{i-1}) \tag{5.1}$$

The Word2Vec method is a kind of simple language model method as it uses surrounding words to predict the middle one, or vice versa, from a window of word. Thus, it computes a probability between words. However, it is built over a "bag of word" technique. Therefore, when the model is trained, the surrounding words (context words) are used without any order consideration. Based on this assumption, modern deep learning-based language models use structures able to learn this order but also the dependencies between words.

5.3.2 The Perplexity Metric

In the NLP domain, language models are trained and evaluated using a particular metric, the "Perpexity" [18]. It indicates the variability of a prediction model. In other words, perplexity is a measurement of how well a probability model predicts a sample, a word in the NLP context. It may be used to compare probability models. A low perplexity indicates the probability distribution is good at predicting the sample. Lower perplexity corresponds to lower entropy and, thus, better performance. The perplexity for a language model can be defined as the inverse probability of the test set, normalised by the number of words, EQ 5.2.

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, ..., w_{i-1})}} \tag{5.2}$$

W is the test set. It contains the sequence of words of all sentences one after the other, including the start-of-sentence and end-of-sentence special words, <SOS> and <EOS>, usually used in the NLP field. For example, a test set with two sentences would look like this:

$$W = (< SOS >, First, sentence, ., < EOS >, < SOS >, Second, one, ., < EOS >)$$

N is the count of all words in the test set, including special words and punctuation. In the example above $N = 10$. It can also be defined as the exponential of the cross-entropy. This last definition is generally used for the model training.

5.3.3 Contextualized Word Embedding: ELMo

Generally, word embeddings, such as Word2Vec, fail to deal efficiently with lexical polysemy. These word embeddings cannot take advantage of information of the context in which the word was used. Instead of using a fixed embedding for each word, ELMo [13] looks at the entire sentence before assigning each word in its embedding. The core idea behind contextual word embeddings is to provide more than one representation for a word, based on the context in which this word appears. ELMo acquires its understanding of language by being trained to: (1) predict the next word in a sequence of words ("language modeling", Eq 5.1), and (2) also predict the previous word ("reverse language modeling", Eq 5.3). This association defines the "bidirectional language modeling" and the two tasks are performed in the same time by the model.

$$P(w_i | w_{i+1}, w_{i+2}, ..., w_N) \tag{5.3}$$

More concretely, ELMo uses a bidirectional LSTM trained on the specific bidirectional language modeling task to be able to create such embeddings. It uses two parallel stacked LSTM to capture information from past and future context to encode the current word (see Figure 5.11).

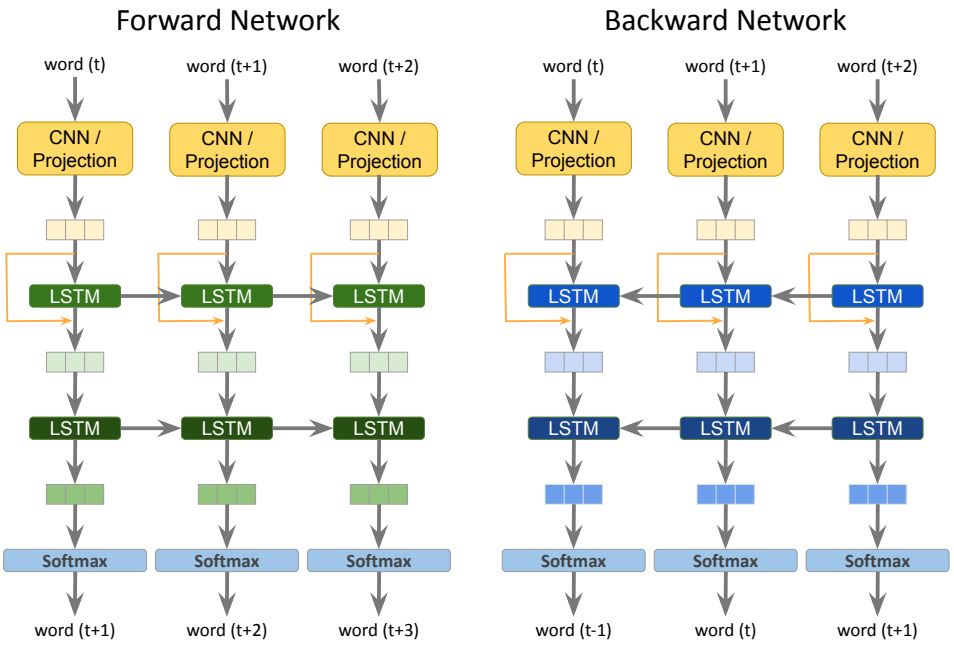


Figure 5.11 – ELMo architecture

The input words are, depending on the version of ELMo, encoded via an embedding layer or a CNN. The CNN input version was proposed, in a second time, in order to treat the unknown words. In this last version, words are seen at the level of characters. Each word character is turned into an id and embedded with a CNN. This CNN embedding learns how words are built.

Then, at each time step, the vector representation of a given word is passed to the forward and backward LSTM layer with residual connections. The outputs from these two LSTMs are further passed to another forward and backward LSTM layer. Finally, there is a softmax layer used to predict the next word from the forward pass network and the previous word from the backward network. In the ELMo model, only the word representation (CNN output or embedding output) and the softmax layers are shared between the forward and the backward networks, which generate a consistent representation of words.

5.3.4 ELMo for Sensor Event Embedding

Based on the ELMo model, we have trained a sensor contextualized embedding. Our objective is to embed a sensor vector activation according to its context, such as words. To do so, we used the first version of the ELMo model (without the CNN layer). This choice was motivated by the

fact that the "smart home vocabular" is less extensive than a real language, so unknown values should be rarer. In addition, the datasets provide a large amount of example sensor activations which should cover all possible variations.

The ELMo model inputs are activity sequences. As activity sequences can be long and in order to accelerate the training of the model, we limit the past and the future context length as it is done in the original ELMo implementation.

As in the NLP domain, two special words, "<start>" and "<end>", are added into the vocabulary to mark the beginning and end of activity sequences. These special words allow us to use the last and the first sensor activation of an activity sequence. In other words, the last sensor activation of the sequence will predict the special word "<end>" in the forward network and the first sensor activation will predict the special word "<start>" in the backward network (see Figure 5.12).

Moreover, as it is usually made in the NLP field we also add to the vocabulary another special word, "<UNK>", to prevent out of vocabulary words. Out of vocabulary words refer to the new words which are encountered at testing. These new words do not exist in the vocabulary. Hence, these methods fail in handling these words. In our case, thanks to this special word, an unknown sensor's activation in the test dataset will be replaced by the word "<UNK>", allowing the model to still encode and represent the sequence.

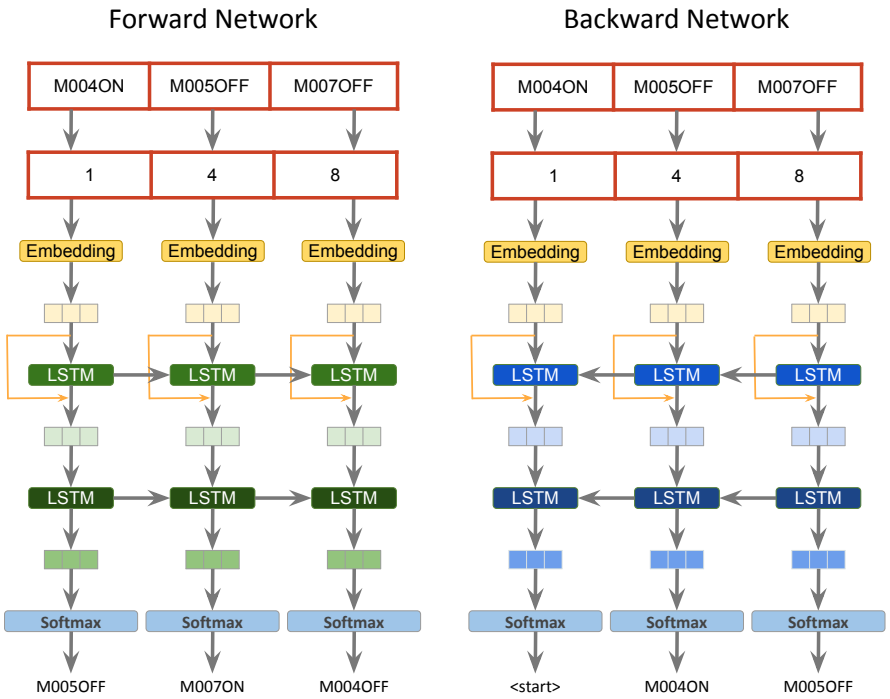


Figure 5.12 – ELMo sensor activation architecture

To train the sensor ELMo version, we respect the same procedure as in the NLP domain for language model generation. The corpus is divided into three subsets for: training (80%), validation (10%) and testing (10%). ELMo uses only the context contained in the sentence. In other words, the context of the previous and following sentences is not taken into account. Therefore, to realize this division, we proceeded as follows. Each sentence is considered independent. This allows to shuffle the whole corpus (set of activity sequences) and to divide it according to the above proportions. Each subset contains a percentage of the activity sequences without order or dependency relations. During training, we used perplexity to track the learning performance of the model in place of accuracy. All ELMo models were trained following the parameters detailed in Table 5.2

Table 5.2 – ELMo embeddings hyperparameters

Embedding size	64
Context windows size	60
Max epochs number	400
Batch size	512

5.3.5 ELMo Sensor: Activity Sequences Embedding Visualization

The main advantage of ELMo is to provide more than one representation for each word based on the context in which it appears. Therefore, as the word vector provided by this embedding depends on surrounding words, it is not appropriate to visualize ELMo embedding of isolated words. However, the visualization of the sentence embedding garners the appearance of some interesting cues.

In order to visualize the embedding of the activity sequences, we proceeded in the same way as for the Sensor2Vec method. A GAP layer is added to the ELMo embedding to generate a unique activity sequence embedded vector (see Figure 5.13). Then, these vectors are reduced via the UMAP algorithm to be displayed into a 2D-dimensional space. Finally, each dot is colored according to the activity label. The Figures 5.14, 5.15 and 5.16 show the activity sequence embedding via ELMo of the Aruba, Milan and Cairo datasets respectively.

Globally, compared to the Sensor2Vec method, the clusters proposed by the association of ELMo and UMAP are more isolated from each other. This seems means that ELMo is able to extract more revealing features than Sensor2Vec. Moreover, within the activity class “Other”, isolated clusters appear. We assume that these clusters of activation sequences, labeled “Other”, are in fact potential new activity classes. ELMo mixed with UMAP seems a possible way to discover new activities. This visualization demonstrates the strong ability of the ELMo embedding to capture relevant features from raw sensors activations.

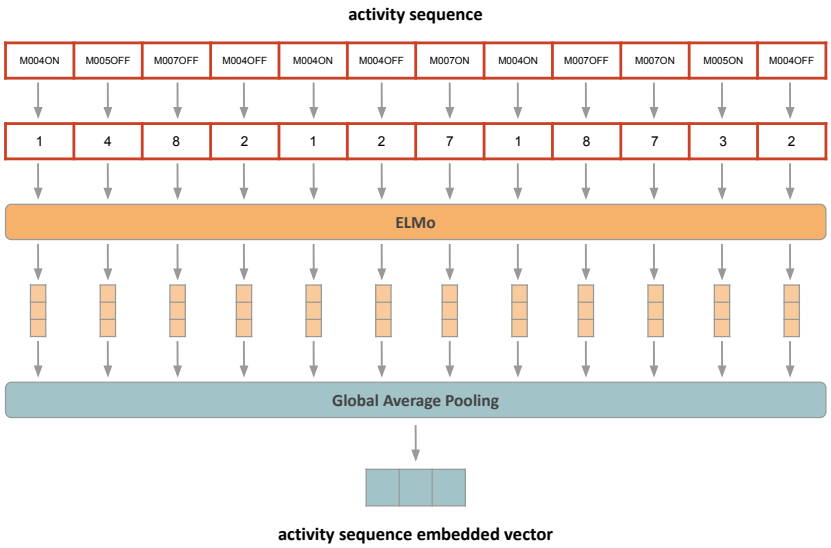


Figure 5.13 – ELMo activity sequence embedding model

However, this clustering is not perfect, when looking closely at the clusters. Some points of a different color than the cluster majority color can appear. This mix can be explained in two ways. First, some sequences belonging to two different classes may be very similar in terms of sensor activation because there is not enough sensor to distinguish them. Second, it is potentially a labeling error in the dataset. Indeed, it is difficult to create and label datasets about human activities in houses [2].

Looking at the Aruba ELMo embedding projection (Figures 5.14) we can observe that sequences that belong to the same class seems well grouped. However, we can note an important class confusion between “Enter Home” and “Leave Home”. This confusion seems normal as these two activities are very similar. Sequence that belong to the “Eat” activity are totally mixed into clusters that belong in majority to the class “Other”. We can observe similar clustering errors on the two other datasets. For Milan “Bathing” and ”Bed to toilet” activities are mixed. We can see the same things for the activities class of “Cook” and ”Take Medicine”. For the dataset Cairo, the “Eat” and ”Take Medicine” activities seem the most mixed. In addition, the ”Bed to toilet” activity is often mixed with the activity “Other”.

The combination of the ELMo pretrained embedding and a dimension reduction algorithm, such as UMAP, seems able to make unsupervised clustering of similar activity sequences. Indeed, the training of the ELMo embedding and UMAP are done in an unsupervised way, i.e., by using no labels. Therefore, it seems possible to group pre-segmented activity sequences completely unsupervised. In other words, if it is possible to split a dataset into unlabeled activity sequences and, by using ELMo, and UMAP it seems possible to group sequences belonging to almost the same class.

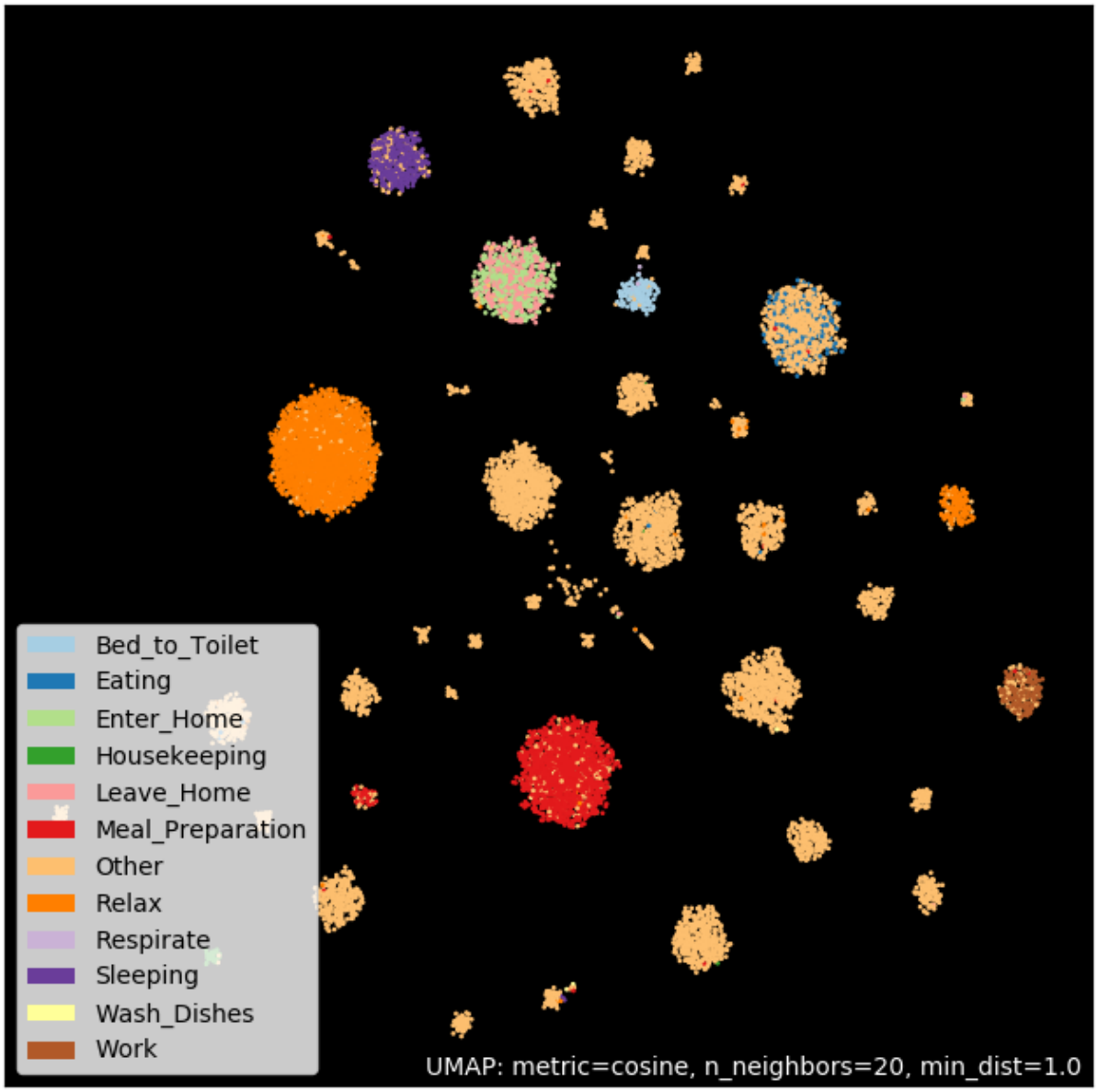


Figure 5.14 – Activity sequences of Aruba embedded by ELMo

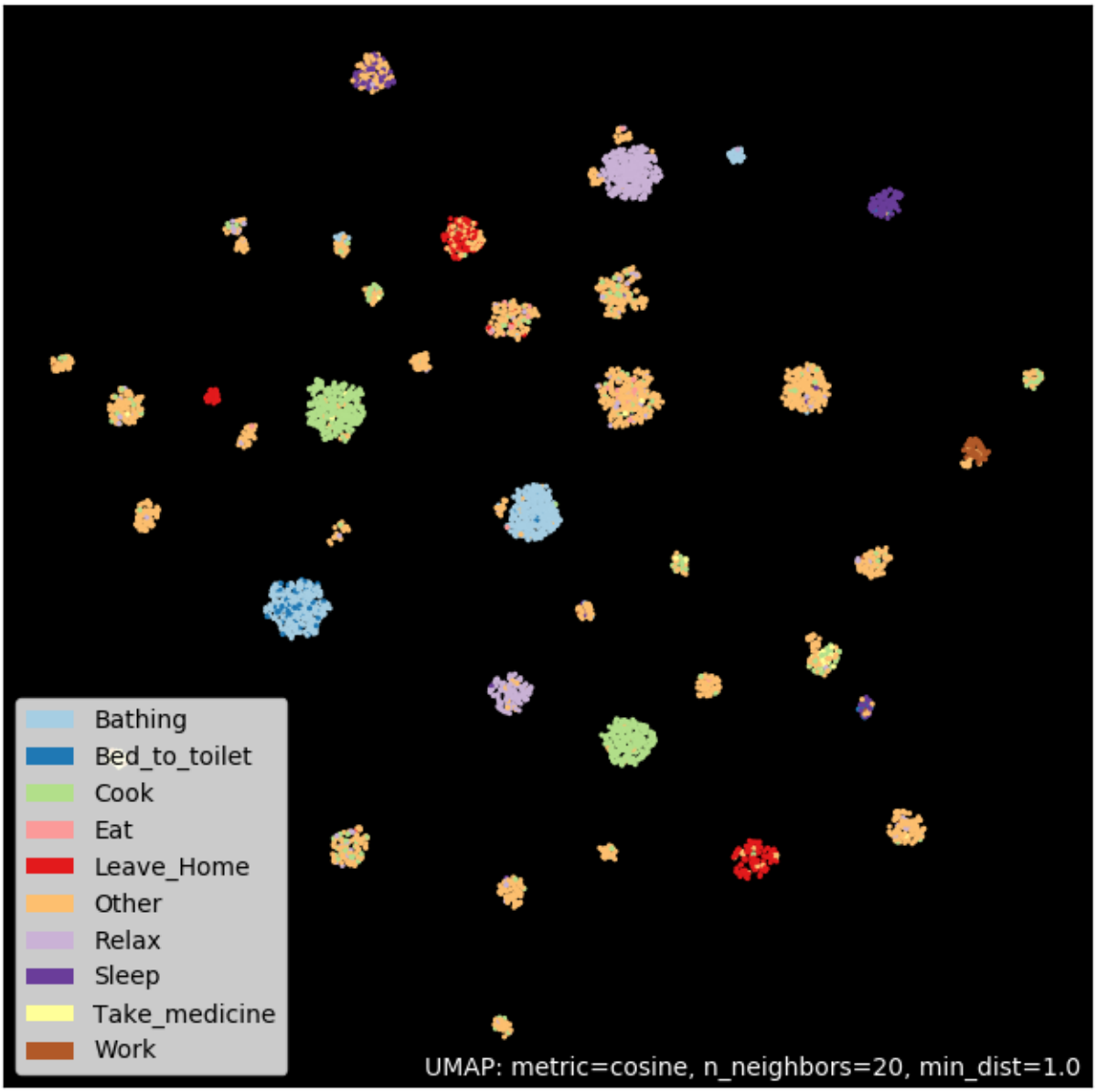


Figure 5.15 – Activity sequences of Milan embedded by ELMo

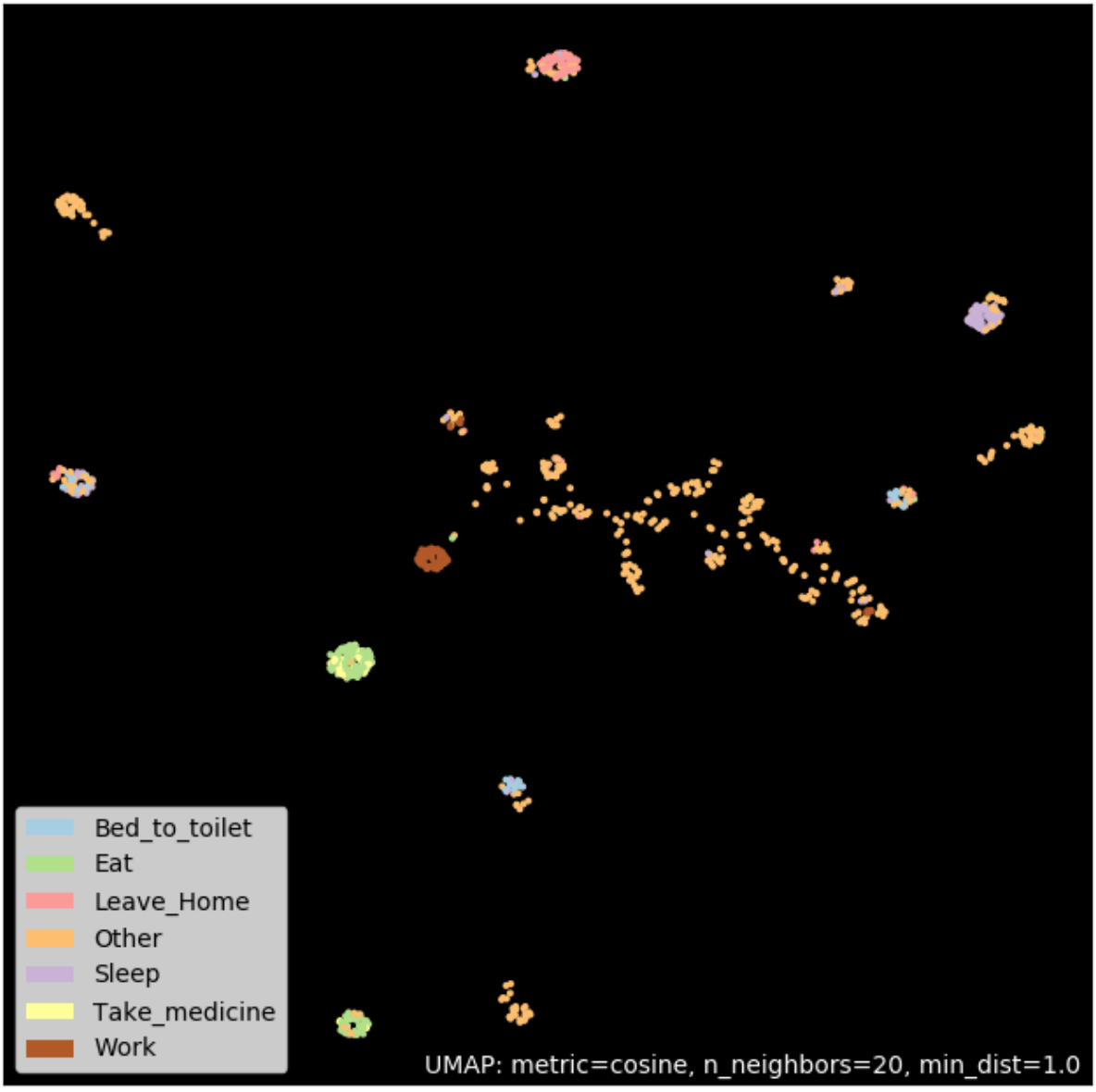
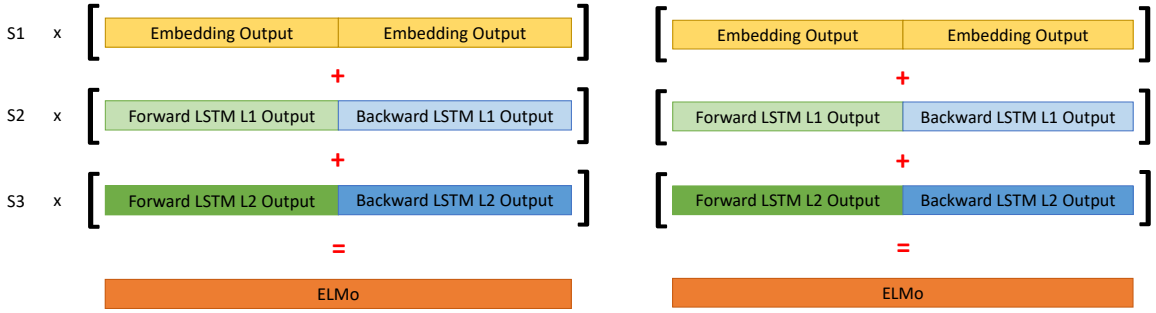


Figure 5.16 – Activity sequences of Cairo embedded by ELMo

5.4 Activity recognition with Pre-trained Embedding

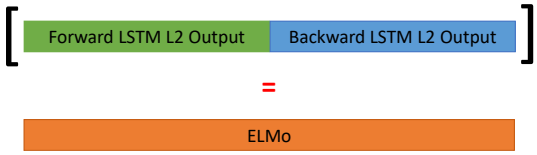
5.4.1 ELMo Outputs Comparison

Usually, the ELMo representation is the weighted sum of the outputs of the different layers level of the model, where the weights are trained according to the final task (see Figure 5.17a). However, three other output forms were evaluated, as depicted in Figure 5.17. First, the sum of outputs by level (Figure 5.17b). Secondly, the last output layer of the model (Figure 5.17c). Finally, the concatenation output of the different layers of the model (Figure 5.17d).

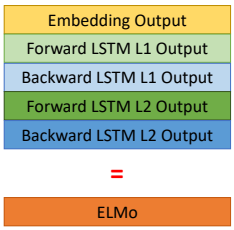


(a) ELMo output: weighted sum

(b) ELMo output: sum



(c) ELMo output: last



(d) ELMo output: concatenate

Figure 5.17 – ELMo outputs.

The Table 5.3, shows the comparative results for the three studied datasets. To perform this experiment, we replaced the embedding layer of the bi-directional LSTM neural network structure proposed by Liciotti et al. [8] with the various ELMo outputs. From the Table 5.3, we can see that the "Concat" output outperforms all others except the Milan dataset in terms of recall, F1-score and balanced accuracy. We assume that this performance comes from the fact that the "Concat" output gives to the classifier the choice to select which feature is the most relevant from each level of representation. However, except the "Last" output version, the "Weighed Sum" and the "Sum" have very close scores. The "Weighted Sum" output has a slight advantage over the "Sum" output. Based on this result all the next experiments use the "Concat" output to train the classifier part.

Table 5.3 – ELMo outputs types comparison

	Aruba				Milan				Cairo			
	Weighted Sum	Sum	Last	Concat	Weighted Sum	Sum	Last	Concat	Weighted Sum	Sum	Last	Concat
Accuracy	96,80	96,77	96,25	**96,86**	89,90	89,74	89,58	**90,25**	88,70	88,52	86,68	**89,36**
Precision	94,50	94,47	93,5	**94,63**	84,00	82,91	84,09	**85,91**	81,00	80,24	81,15	**84,30**
Recall	88,60	88,4	87,59	**88,73**	**76,00**	75,93	75,05	74,85	78,30	76,31	78,06	**80,21**
F1-score	90,50	90,46	89,59	**90,71**	**78,70**	78,54	78,34	77,86	78,50	77,78	79,02	**81,34**
Balanced Accuracy	88,56	88,39	87,59	**88,72**	**76,00**	75,92	75,04	74,84	77,00	76,3	78,05	**80,20**
Weighted Precision	96,90	96,89	96,35	**96,98**	89,80	89,43	89,38	**90,06**	88,05	87,64	86,77	**89,24**
Weighted Recall	96,82	96,77	96,25	**96,86**	89,80	89,74	89,58	**90,25**	88,80	88,52	86,68	**89,36**
Weighted F1-score	96,72	96,7	96,18	**96,79**	89,60	89,43	89,26	**89,74**	88,10	87,88	86,45	**89,02**

5.4.2 Comparison with Context-Free Embeddings

To evaluate our proposed methods using pre-trained embbeding, we compare them to the baseline models define in Chapter 3, which does not use contextual information. We also add two models for comparison, a LSTM and a bidirectional LSTM without embedding layer, to validate the improvement of this layer.

The experiment results in Tables 5.4 and 5.5 show that the bidirectional LSTM obtained better results, as shown in reference [8]. In addition, using an embedding layer improved the global classification performance, thanks to the ability of this layer to capture meaningful features between sensors' activations. This last confirm the utility of using a projection layer.

Against all odds, we notice that the Sensor2Vec embedding did not improve the classification performance compared to baseline models. On the contrary, using Sensor2Vec decreases the performance. This seems means that the Sensor2Vec embedding did not accurately capture some important features by itself. The knowledge of some sensors semantic and localisation seems not enough.

The ELMo embedding performs better than all other methods on every dataset except on the Milan dataset but shows a high improvement on the multi-user dataset Cairo. It increases the F1-score and the Balanced Accuracy by 10 points. Multi-user dataset seems requires the understanding of the context in order to differentiate users' activities. For the Milan dataset ELMo is slightly behind the Liciotti-baseline model, but ELMo win in precision. Concerning the training time, we noted a negligible gain of about 30 s to 2 min, depending on the dataset, by using a pretrained embedding.

In order to evaluate if the ELMo model could improve a less efficient classifier, we also evaluated the ELMo embedding coupled with the FCN studied in Chapter 4. For this experiment we used the FCN with the circular padding as it demonstrated the best performances. The experiment results in Table 5.6 show that the FCN combined with ELMo help to reduce the gap between the standard FCN model and the Liciotti-baseline model. One again the most important improvement can be observed over the Cairo dataset.

Table 5.4 – Without and with embedding coupled with a LSTM Classifier.

	Aruba				**Milan**				**Cairo**			
	No Embedding	Liciotti	Sensor2Vec	ELMo	No Embedding	Liciotti	Sensor2Vec	ELMo	No Embedding	Liciotti	Sensor2Vec	ELMo
Accuracy	94.63	96.50	96.6	**96.61**	79.18	**89.88**	86.31	87.29	74.82	81.44	79.31	**82.62**
Precision	71.84	80.50	79.39	**86.24**	76.22	79.17	81.75	**86.16**	60.73	68.88	55.31	**78.10**
Recall	70.00	79.52	**80.07**	79.39	61.89	74.26	69.33	**75.35**	60.46	69.98	62.90	**72.358**
F1 score	70.55	79.49	79.34	**80.73**	65.94	76.28	72.07	**78.90**	59.11	68.80	58.92	**73.78**
Balance Accuracy	70.00	79.52	**80.11**	79.39	61.88	74.26	69.33	**75.35**	60.46	69.98	62.91	**72.58**
Weighted Precision	93.56	96.11	96.21	**96.41**	78.84	**88.51**	86.08	87.26	72.08	79.64	73.6	**82.77**
Weighted Recall	94.06	95.5	96.6	**96.61**	79.18	**88.99**	86.31	87.29	74.82	81.44	79.31	**82.62**
Weighted F1-score	93.77	**96.88**	96.32	96.43	78.42	**88.63**	85.79	87.07	72.73	80.09	75.79	**82.05**

Table 5.5 – Without and with embedding coupled with a Bi-LSTM Classifier.

	Aruba				Milan				Cairo			
	No Embedding	Liciotti	Sensor2Vec	ELMo	No Embedding	Liciotti	Sensor2Vec	ELMo	No Embedding	Liciotti	Sensor2Vec	ELMo
Accuracy	95.01	96,61	96.59	**96,86**	82.24	**90,82**	88.33	90,25	81.68	85,33	82.27	**89,36**
Precision	79.75	92,5	82.97	**94,63**	79.6	79,94	84.42	**85,91**	68.45	74,49	77.56	**84,3**
Recall	77.73	88,68	81.06	**88,73**	67.77	**74,91**	73.62	74,85	70.09	69,86	69.38	**80,21**
F1-score	77.92	89,92	81.43	**90,71**	71.81	**76,86**	76.59	**77,86**	68.47	71,1	70.95	**81,34**
Balance Accuracy	77.73	88,67	81.06	**88,72**	67.77	**74,90**	73.61	74,84	70.09	69,85	69.38	**80,2**
Weighted Precision	94.69	96,73	96.23	**96,98**	82.28	**90,31**	88.28	90,06	80.22	84,29	82.04	**89,24**
Weighted Recall	95.01	96,61	96.59	**96,86**	82.24	**90,82**	88.33	90,25	81.68	85,33	82.27	**89,36**
Weighted F1-score	94.74	96,55	96.32	**96,79**	81.97	**90,41**	87.98	89,74	80.49	84,43	81.14	**89,02**

Table 5.6 – Liciotti Bi-LSTM and ELMo + Bi-LSTM and ELMo + FCN(Circular Padding) comparison

	Aruba				Milan				Cairo			
	Liciotti	ELMo	ELMo+ FCN	FCN	Liciotti	ELMo	ELMo+ FCN	FCN	Liciotti	ELMo	ELMo+ FCN	FCN
Accuracy	96,61	**96,86**	94.59	94,09	**90,82**	90,25	85.33	80,12	85,33	**89,36**	84.39	53,3
Precision	92,5	**94,63**	73.23	77,86	79,94	**85,91**	83.74	65,07	74,49	**84,3**	79.28	21,29
Recall	88,68	**88,73**	69.27	73,73	**74,91**	74,85	70.49	66,48	69,86	**80,21**	80.08	14,19
F1-score	89,92	**90,71**	69.30	73,84	**76,86**	**77,86**	74.79	64,62	71,1	**81,34**	78.13	13,67
Balance Accuracy	88,67	**88,72**	69.57	72,47	**74,90**	74,84	70.86	65,11	69,85	**80,2**	80.29	17.15
Weighted Precision	96,73	**96,98**	94.09	93,61	**90,31**	90,06	85.31	80,35	84,29	**89,24**	83.91	35.01
Weighted Recall	96,61	**96,86**	94.53	93,93	**90,82**	90,25	85.36	79,87	85,33	**89,36**	82.27	51.83
Weighted F1-score	96,55	**96,79**	94.10	93,58	**90,41**	89,74	85.02	79,41	84,43	**89,02**	82.27	37.56

5.4.3 Comparison Against Staked Bidirectional LSTM

The ELMo model can be approximated by a bidirectional LSTM layer with an embedding layer. In this experiment, we compare the ELMo method to two stacked layers of bidirectional LSTM with an embedding layer. Table 5.7 shows the results of this comparison.

Table 5.7 – Comparison with two layers of Bi-LSTM.

	Aruba			Milan			Cairo		
	1L	2L	ELMo	1L	2L	ELMo	1L	2L	ELMo
Accuracy	96,61	96.46	**96,86**	**90,82**	90.03	90,25	85,33	84.99	**89,36**
Precision	92,5	82.01	**94,63**	79,94	84.29	**85,91**	74,49	80.87	**84,30**
Recall	88,68	79.05	**88,73**	**74,91**	77.31	74,85	69,86	76.6	**80,21**
F1-score	89,92	79.97	**90,71**	76,86	79.29	**77,86**	71,1	77.44	**81,34**
Balance Accuracy	88,67	78.74	**88,72**	**74,90**	75.51	74,84	69,85	76.52	**80,20**
Weighted Precision	96,73	96.04	**96,98**	**90,31**	90.22	90,06	84,29	85.04	**89,24**
Weighted Recall	96,61	96.41	**96,86**	**90,82**	90.28	90,25	85,33	84.4	**89,36**
Weight F1-score	96,55	96.13	**96,79**	**90,41**	90.07	89,74	84,43	84.08	**89,02**

The results demonstrate that the ELMo structure obtains a non-negligible gain on the three datasets, except on the Milan dataset. On this one, gains are less, but not negligible, on the Precision and F1-score. ELMo still allows recognizing more classes and is more precise than the other structures. The stacking of bidirectional LSTM does not allow obtaining better performance. This type of structure can even degrade the performance in some cases, as on the Cairo dataset, or Aruba. It seems that the method to train ELMo helps to capture more useful features.

5.4.4 Confusion Matrices Analysis

The confusion matrices in Appendix C presents the confusion and classification scores activities for the three-folds experiments over the three studied datasets for the Liciotti-baseline (embedding + bidirectional LSTM) and ELMo + bidirectional LSTM approaches. We observe that the misclassification rate (1 - accuracy score) is lower with ELMo method in general except over the Milan dataset.

The performance of both models are very close for the dataset Aruba. We can observe more differences for the dataset Milan. Depending on the folds, the Liccioni-baseline model seems to perform a little bit better than the ELMo approach. However it seems that the ELMo model can recognize better the activity “Eat” 2 folds over 3.

Concerning the dataset Cairo, the ELMo model confirms its performance. It recognizes all activity classes independently of the fold. Moreover, the ELMo model shows a better percentage of recognition in classes.

However, these matrices show a simplified version of the problem. Indeed, we have grouped some activities under a “meta” label in order to compare fairer the datasets between them. Thus, we retrained models over the three datasets according to their original activity label and results are a little bit different (see confusion matrices in Appendix D).

For the Aruba dataset, we observe that the medical activity “Respirate” is not correctly recognized for the both methods. This activity has very few examples (5 examples) and, therefore, is difficult to recognize. The ELMo approach demonstrates the ability to recognize this activity over the three folds while the Liciotti-baseline model recognized it in only one fold.

The activity “Wash Dishes” is poorly recognized and is completely confused with the activity “Meal Preparation”. This mixing is explained by the fact that these two activities activate the same sensors. This misclassification was already observable on the previous Aruba’s confusion matrices where a small part of the “Work” activity is mixed with the “Cook” activity. In order to differentiate these two activities, the algorithms should be able to take into account the time of day, or the activity preceding the current activity.

In the case of the Milan dataset principally three activities seems poorly recognized (“Chores”, “Dining_Rm_Activity” and “Eve_Meds”). The activity “Dining_Rm_Activity” is poorly recognized because it mainly activates only the motion sensor located in the dining room, which is often activated during resident travel between the living room and the kitchen. However, the ELMo approach is able to find a part of the occurrences of this activity. This performance is explained by the fact that ELMo captures in which context the motion sensor in the dining room is activated. In other words, it is able to differentiate between the resident’s passage in the dining room to join the kitchen or the living room, from a real activation of the sensor for this activity. The “Chores” activity is very similar to the activity “Other” as during this activity the resident moves in all the house and activates random sensors. The activity “Eve_Meds” is mainly mixed with the activity “Morning_Meds” which seems rational as it is the same activity but at two particular moments of the day.

Cario’s confusion matrices show the difficulty of both models to correctly identify the activities that belong to the "Cooking" activity (“Breakfast”, “Dinner” and “Lunch”). Indeed, these three activities depend on the time of the day. We can see that the ELMo approach is able to almost distinguish them. Moreover, ELMo demonstrates to in general better recognize all activities.

5.5 Pre-Trained Embedding: From One House to Another

5.5.1 Help Classifiers with Embedding Transfer Learning

In the field of NLP, embeddings are pretrained on large corpora and then used on a more specific corpus to perform particular tasks, such as text classification. This is the principle of transfer learning. The objective is to use the very generic features of the large corpus on a smaller and specific corpus to gain in learning time but also in genericity. In the case of ADLs recognition, this practice would allow for transfer the knowledge from a smart house to another one so that the latter can recognize ADLs without further training. This transferred model could then be refined for the context of this new house. We experimented with this practice using Aruba's ELMo embedding on the Cairo and Milan datasets.

To evaluate the transfer learning ability of a ELMo pre-trained sensor embedding we exploited the same previous model structure, with the exception that we use the ELMo embedding of Aruba for the training of classifiers. In this model, the Aruba ELMo embedding extract and encode the activity sequence of the Cairo or Milan datasets. Then the output representation of the embedding is used by the classifier. The classifier extract patterns and features to perform the final task. During training, the weights of the ELMo embedding were frozen, and only the bidirectional LSTM and the softmax were trained to classify the activities of the second dataset.

Table 5.8 – ELMo from Aruba applied to Cairo and Milan VS original ELMo and Liciotti-baseline

	Cairo			Milan		
	ELMo from Cairo	ELMo from Aruba	Liciotti	ELMo from Milan	ELMo from Aruba	Liciotti
Accuracy	**89,36**	88,1	85,33	90,25	90,36	**90,82**
Precision	**84,3**	79,39	74,49	85,91	**87,82**	79,94
Recall	**80,21**	73,44	69,86	74,85	**75,61**	74,91
F1-score	**81,34**	75,61	71,1	77,86	**79,52**	76,86
Balance Accuracy	**80,2**	73.43	69,85	74,84	**75,6**	74,90
Weighted Precision	**89,24**	87,1	84,29	90,06	90,29	**90,31**
Weighted Recall	**89,36**	88,1	85,33	90,25	90,36	**90,82**
Weighted F1-score	**89,02**	87,29	84,43	89,74	90,03	**90,41**

Looking at the result in Table 5.8, the generic features learned from the Aruba dataset allowed the classification of activities in the Cairo dataset with scores a little bit less, but close, compared to the ELMo embedding trained on Cairo. Nevertheless, the ELMo from Aruba outperform the Liccioti-baseline which is train on the target data. We can observe on contrary on the Milan dataset that using the ELMo from Aruba improve sightly results compared to the ELMo originally trained on Milan. We can also see that over this dataset the ELMo from Aruba-based model shows better performance against the Liciotti-baseline in term of precision, recall and F1-score.

But, why the embedding transfer learning works ? Indeed, the two target datasets do not have the same number of sensors and can have different sensors values, in particular for the temperature sensors, compared to Aruba. However, the three datasets belong to the CASAS benchmark datasets, which use the same sensor type and the same denomination structure. The sensors follow the structure: "sensor type" + index. Moreover, Aruba have a larger vocabulary which theoretically avoid out of vocabulary words problem. The Aruba vocabulary have a size of 259 possible words, while Milan have 136 different words, and Cairo have 197 possible words. Nevertheless, the house typologies are completely different, and sensors with the same names not sharing the same location or function between datasets. In other words, a sensor with the name ID "M019" which is a kitchen motion sensor into Aruba is motion sensor for the bedroom door in Milan and a motion sensor for the kitchen cabinet in Cairo.

However, we observe that, even if a word does not have the same "meaning" from one dataset to another the encoding provided by the ELMo embedding generate patterns still allows the classifier to reach performance rates equivalent to a model fully trained on the destination dataset. We conjecture this good performance comes from the fact that ELMo takes into account the word order and generate a similar "syntax" for activity sequence that belong to same classes. Thus, even if the input words have changed, the syntax is captured by the classification part of the model which compensate the gap, i.e., the order of each word, as well as the patterns of their recurrence in the sequence. These results indicate the importance of contextualized embeddings.

5.5.2 Embedding Output Analysis

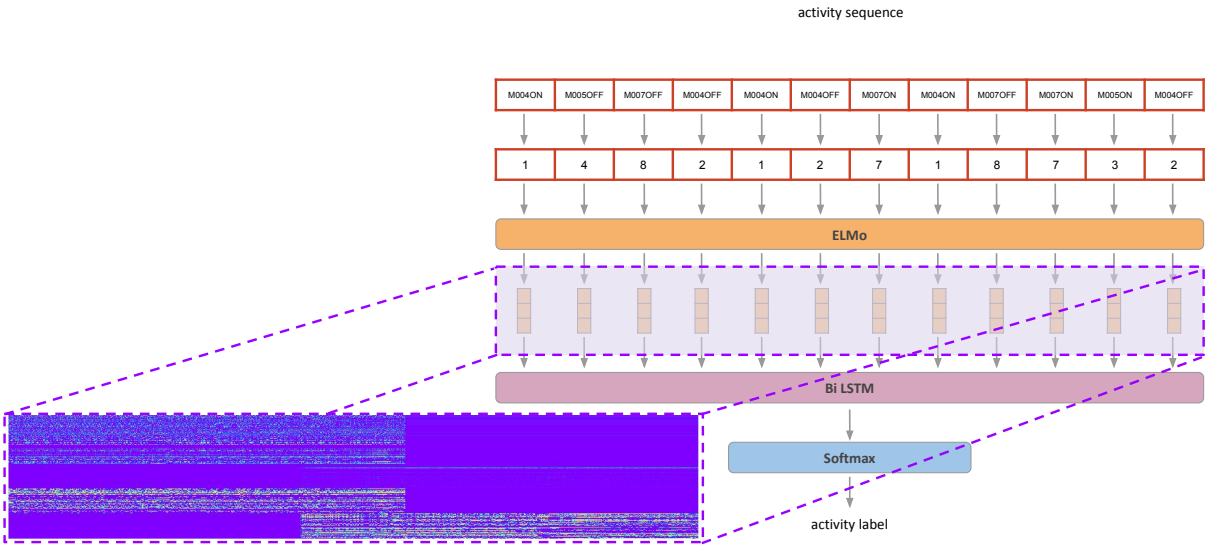


Figure 5.18 – ELMo output vizualisation

One of our suppositions in the success in transfer learning of the ELMo embedding is that the embedding provides a similar output for activity sequences belonging to the same class. In order to try to validate this hypothesis we search to create a heat-map of the embedding layer output to expect to observe visual patterns, see Figure 5.18. We use it as input of the Aruba embedding Cario's activity sequences. The output of two random "Eat" activity sequences can be seen in Figure 5.19.

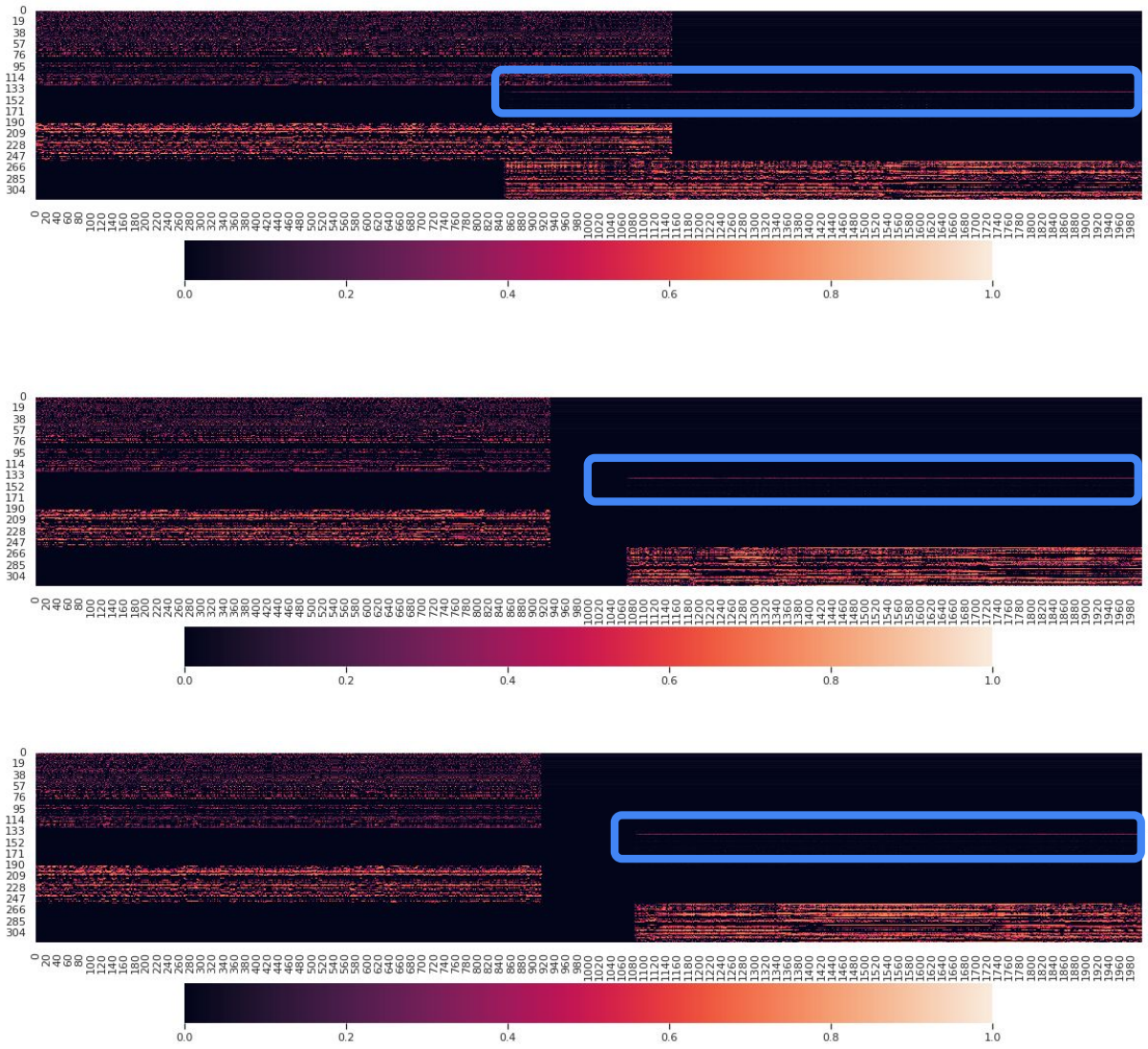


Figure 5.19 – ELMo output heat-map of two random Cairo's "Eat" activity sequences

The three extracted "Eat" activities look very similar, and seem to have the same patterns. In particular we can observe exactly the same straight line (blue rectangle) between the three extracts. The Figure 5.20 compare a "Eat" activity heat-map and "Sleep". At a first look except

length patterns they appear similar, but if we zoom in at the end of the sequence pattern we can see some differences. Contrary to the activity “Eat” the activity “Sleep” has in addition to the line, other small lines and dots.

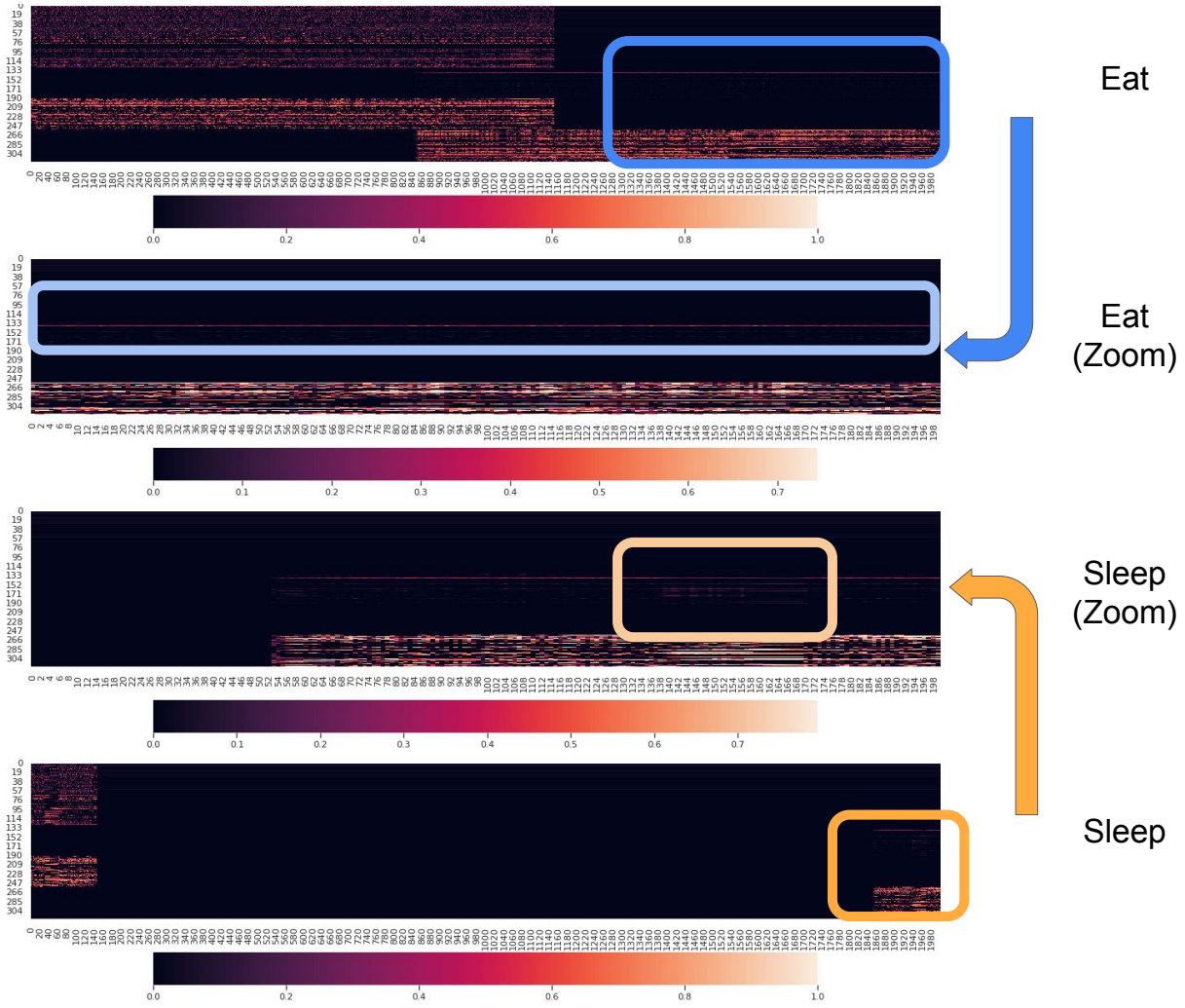


Figure 5.20 – “Eat” and “Sleep” heat-maps comparison

It seems that particular patterns belong to activity classes. Nevertheless, the observation of these patterns is not always obvious. Even if our hypothesis seems to be validated, it would be interesting to be able to evaluate or use more adapted and advanced methods of explicability [15].

5.6 Summary

Human activity recognition in smart homes is not only a problem of pattern recognition but also a temporal sequence analysis problem, where the semantics and context of each sensor trigger can change the meaning of a sensor activation. Moreover, the nature of the activated sensor can give a certain amount of information about the current activity.

In this chapter, we proposed a new approach to the field of recognition of ADLs in smart homes. We used techniques from the domain of NLP to capture the context and the semantics of sensor activations in an embedding. This approach allows recognition of a larger number of activity classes, despite the fact that datasets remain unbalanced. Indeed, fewer activation sequences are confused with the "Other" class, which, nevertheless, represents more than 50% of the datasets.

The visualization of the Sensor2Vec embedding allowed us to realize that this method has captured some relations between the activations of sensors. It appears that sensors of the same nature have a close distance in this embedding space. Moreover, the clusters that appear in the space represent different rooms in the house. Nevertheless, this semantic embedding does not provide better performance than the Liciotti-baseline models.

However, our experimentation shows that capturing the context of a sensor activation allows for improvement of the classification of activity sequences, particularly on datasets containing activities performed by several residents. Moreover, we have seen that the contextual ELMo embedding allows a less efficient classifier, especially in our case the FCN, to gains drastically in performance.

Finally, we were able to evaluate that an embedding trained in a house seems to be reused in a new environment containing another denomination of sensors and allow a high rate of classification of activities. It should be noted that these methods are capable of extracting generic information, transferable to other datasets. This last observation suggests that transfer learning between environments is possible through these methods, as it is possible today in the NLP domain.

Through our proposition, that combines a sensor activations model based on language model with a sequence model, our experimental results on real smart home data highlight the importance of a dynamic contextualized semantic representation in ADL recognition. However, we have seen with the original datasets labels, that some activities are not correctly recognized. These activities seem to rely on the time of the day or to potential previous activities. In the next chapter we propose to explore the idea to improve ADLs recognition algorithm thanks to time information or previous activities context.

BIBLIOGRAPHY

[1] ES Abramova, KV Makarov, and AA Orlov. Method for undefined complex human activity recognition. In *2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 797–801. IEEE, 2021.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] Hancheng Cao, Fengli Xu, Jagan Sankaranarayanan, Yong Li, and Hanan Samet. Habit2vec: Trajectory semantic embedding for living pattern recognition in population. *IEEE Transactions on Mobile Computing*, 19(5):1096–1108, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4): 150, 2019.

[6] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.

[7] Qian Li, Hao Peng, Jianxin Li, Congyin Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *arXiv e-prints*, pages arXiv–2008, 2020.

[8] Daniele Liciotti, Michele Bernardini, Luca Romeo, and Emanuele Frontoni. A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 04 2019. doi: 10.1016/j.neucom.2018.10.104.

[9] Moe Matsuki, Paula Lago, and Sozo Inoue. Characterizing word embeddings for zero-shot sensor-based human activity recognition. *Sensors*, 19(22):5043, 2019.

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

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[13] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[15] Atefeh Shahroudnejad. A survey on understanding, visualizations, and explanation of deep neural networks. *arXiv preprint arXiv:2102.01792*, 2021.

[16] Koichi Shimoda, Akihito Taya, and Yoshito Tobe. Combining public machine learning models by using word embedding for human activity recognition. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 2–7. IEEE, 2021.

[17] Kushal Singla and Joy Bose. Iot2vec: Identification of similar iot devices via activity footprints. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 198–203. IEEE, 2018.

[18] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA transactions on signal and information processing*, 8, 2019.

Chapter 6

Taking Advantage of Dependence on Previous Activities and Time of the Day

6.1 Introduction

We have seen in the previous chapters that human activity recognition implies a series of complex modeling. Previous works consider human activities as sequences of sensor activations which reflect the actions of residents. They do not take into account temporal information nor do they focus on the recognition of meta activities such as “Cooking”, “Eating”, “Sleeping”, etc., that may be useful for some health monitoring applications [18, 19]. Furthermore, it also appears to be useful for systems to recognize more particular activities such as “Preparing Breakfast”, “Eating Dinner”, “Morning Medication”, “Evening Medication”, etc. Indeed, precision offers the possibility to create specific services or scenarios, as for instance, reminders for morning and evening medication intake. In addition, taking into account time information might improve the recognition of meta activities.

The consideration of relations between activities is another interesting task. Throughout the day, we perform activities that are often linked. For example, “Washing Dishes” is linked to “Cooking” or “Eating” because the activity “Washing Dishes” is generally performed just before or after “Cooking” and “Eating”. Being able to interpret these relations could help systems recognize the current activity, especially when similarities between sensor activation sequences lead to confusions. For example, “Washing Dishes” can be confused with the activity “Cooking” because they are performed in the same room and therefore may activate the same sensors. Knowing that the previous activity was “Eating”, could remove this confusion. The recognition of the activity does not only depend on a local context, but also mainly on a more global context of the activity.

We have seen that LSTM-based models are suitable structures for the human activity recognition problem in smart homes. They are able to consider the time through the timesteps interpretation, but proposed models do not use the temporal evolution along the day. In addition, the

context for the LSTM models consists of the current sequence and part of the training batch, but sequences in the batches are not necessarily correlated, as it is preferable to shuffle the sequences to help models generalization. Nevertheless, few deep learning works have attempted to exploit time or previous activities' information, [5, 4, 1]. For example, Tan et al. [5] have encoded activities sequences into images and used a pixels color line to encode the previous activity label, in order to help the model to recognize the ongoing activity. Then they used a CNN-based architecture for the classification and not a LSTM-based one. In another example, Ramos et al. [4] have encoded a time feature evolution under sine and cosine values and concatenated this feature with the input sensor vector, to allow a LSTM model to use the time of the current activity window. Devanne et al. [1] uses two layers of LSTM to recognise activities of daily living in videos, by concatenating the previous label to the input features.

In this chapter, we propose to improve the previously shown method (the Liciotti et al. [3]'s bidirectional LSTM) and our previous presented method ELMo embedding based. We have done this by using temporal information or pieces of extra-activity information from a more extended context. We firstly provided a brief recall description of the previously seen two models used for baseline. Then in the next three sections, we detail and investigate each proposed extension against the baseline. In the fifth section, we will make a global comparison of the proposed improvement approaches. Finally, we will conclude a summary detailing the main elements of this contribution.

6.2 Methods and Baselines

6.2.1 Methods

Two experiments are conducted to evaluate baseline models and the proposed extensions with the three studied datasets (Aruba, Milan, Cairo). In the first, similar to the experiments in previous chapters, we grouped dataset activities into 11 ADLs, (cf. Chapter 3), in order to make a consistent comparison of the results between datasets. In the second experiment, we used the original datasets' class labels, due to the fact that they are more precise. Moreover, these activity labels could be similar but not realised at the same moment, such as "Breakfast", "Lunch" and "Dinner". This second experiment aims at evaluating models' capacities to recognise a larger number of low-level activities, similar activities, or activities performed at particular moments during the day.

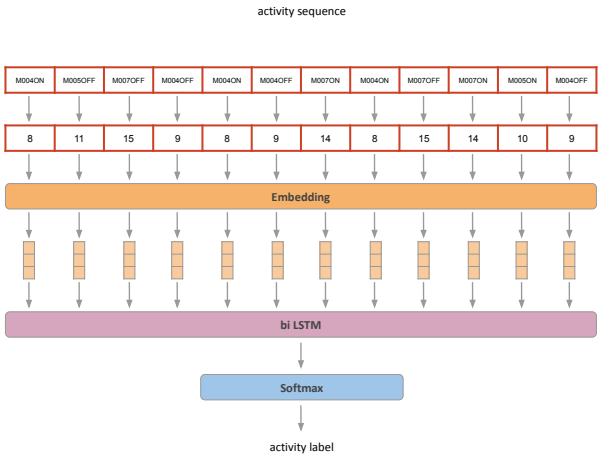
To compare the models, we primarily examine the accuracy and F1-score. We also look at the Balanced Accuracy and weighted F1-score. As a reminder, the first two will guide us on how the models perform on the datasets overall. The second two are studied because the datasets are unbalanced, they will inform us if the models find all activity classes with the same performance or if they focus on the classes with the highest number of occurrences (cf. Chapter 3).

6.2.2 Baselines

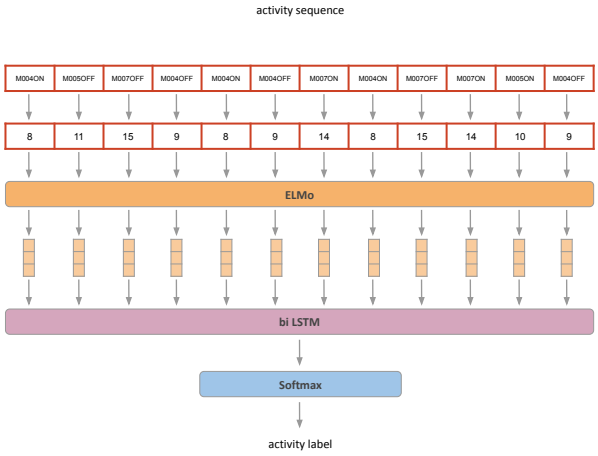
For baseline models, we used :

1. the bidirectional LSTM-based model of [3] (see Figure 6.1a). Henceforward, we note this baseline LB for Liciotti-*baseline*.

2. our proposed model using an ELMo-like pretrained embedding defined in the previous chapter (see Figure 6.1b). We'll use the abreviation EB for ELMo-*baseline*.



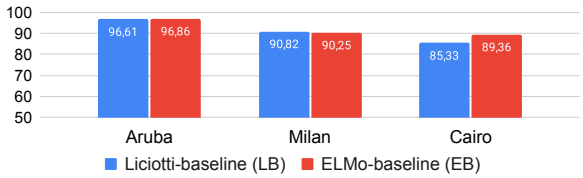
(a) Liciotti-*baseline* (LB)



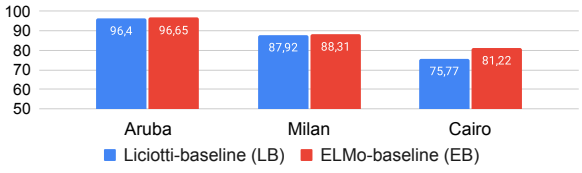
(b) ELMo-*baseline* (EB)

Figure 6.1 – Baseline models

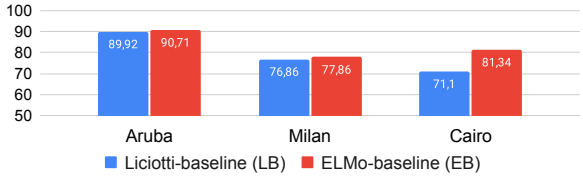
The Figure 6.2 presents, Accuracy, Balanced Accuracy, F1-Score and Weighted F1-Score for the three datasets with grouped labels and original labels. We can observe firstly that scores are lower with original dataset labels. This could be explained by the fact that similar activities, e.g. “Breakfast”, “Lunch” and “Dinner” must be recognized individually while they are packed into the same label in the “Grouped” activity dataset version. We can notice that the EB model performs better than the LB model on all datasets, independently of the dataset version. In addition, the EB model have the best improvement, compared to the LB model, on the Cairo dataset for both datasets versions. Nevertheless, for the grouped Milan dataset version, the LB model seems to perform slightly better than the EB model in terms of Balanced Accuracy and F1-score.



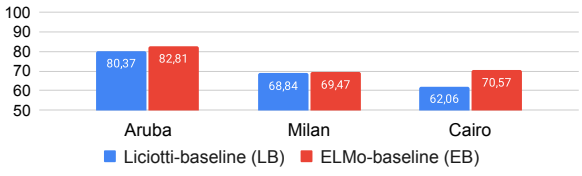
(a) Grouped activities Accuracy



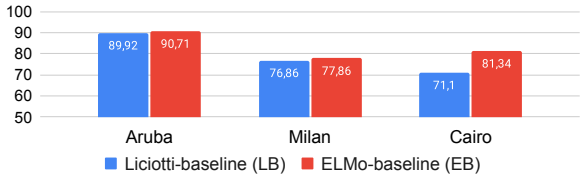
(b) Original activities Accuracy



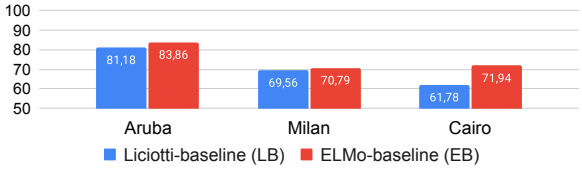
(c) Grouped activities Balanced Accuracy



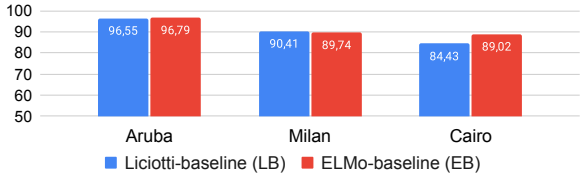
(d) Original activities Balanced Accuracy



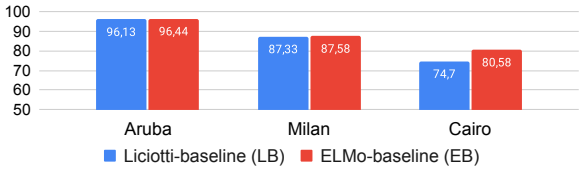
(e) Grouped activities F1-scores



(f) Original activities F1-scores



(g) Grouped activities Weighted F1-scores



(h) Original activities Weighted F1-scores

Figure 6.2 – Scores for baseline models on Aruba, Milan and Cairo datasets

6.3 Previous Activities as an Extension of the Context

As successive activities are interrelated, they are poorly modelled by Markov processes. The information about what the person did before seems useful to recognize the current activity. To capture these relations, we propose to add to the current activity sequence, the two previous activity sequences in input, as in Figure 6.3. In the selected datasets, activity sequences are generally separated by sequences labeled as "Other". The latter corresponds either to unknown activities or to transitions between two activities. That's why we chose to add at least two and not one previous activity sequence. We will note LPA for Liciotti + *previous activities* and EPA for ELMo + *previous activities*.

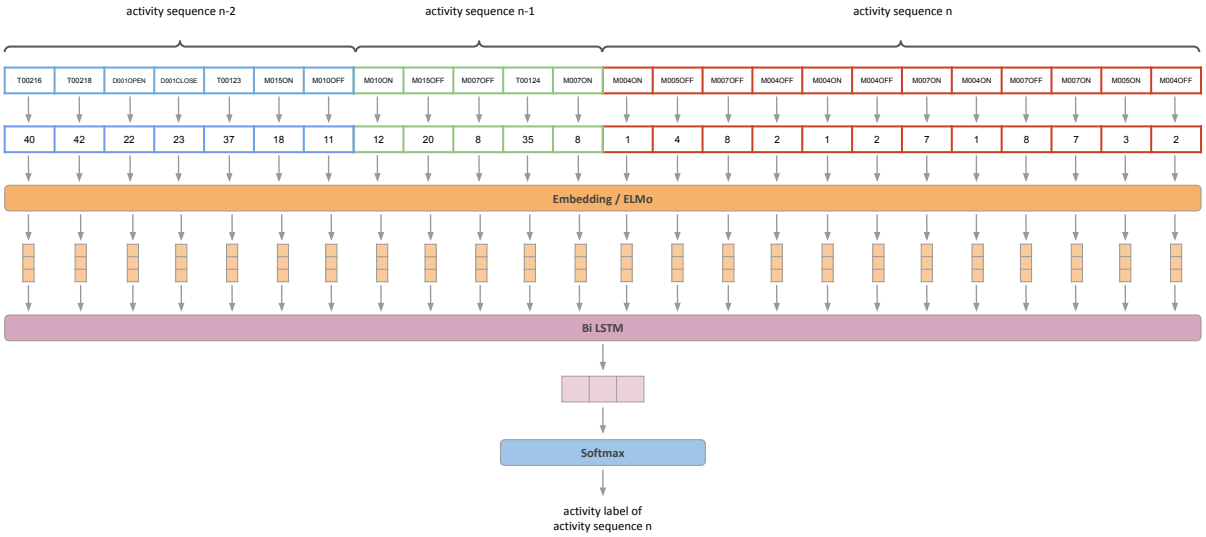
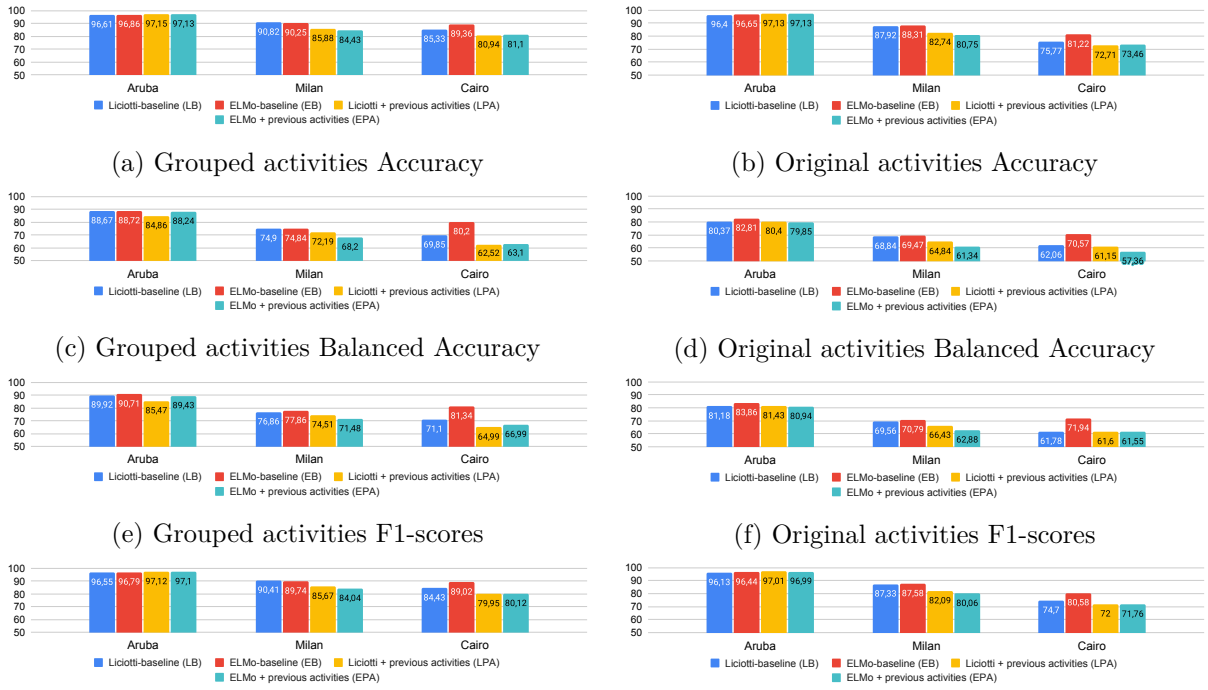


Figure 6.3 – Architecture of ELMo with previous activities

The Figure 6.4 presents comparison metric results of baseline and previous activities models, for both datasets versions. We can observe that this proposed extension (LPA and EPA) decreases the classification performances for both datasets versions. It seems that the knowledge provided by the extended context does not help. The first possible explanation is that sequences become most similar by the addition of previous activities. Another factor could be the sequence length. In our example, the average sequence length increases from ~ 600 to ~ 1800 elements. Yet, a well known problem on long sequences with recurrent neural networks, in part decreased by LSTM, is the vanishing gradient problem [2] that implies that long-term relationships are hardly encoded by recurrent neural networks. The longer the sequence, the harder it is to find time relationships. Moreover, originally, activity sequences had different lengths and activities belonging to the same class had a similar size. For example, the activity "Bed to Toilet" is a very short activity in terms of event numbers compared to "Sleep". It appears that this length information that can be used by the LSTM is destroyed by the addition of previous sequences.



(a) Grouped activities Accuracy

(b) Original activities Accuracy

(c) Grouped activities Balanced Accuracy

(d) Original activities Balanced Accuracy

(e) Grouped activities F1-scores

(f) Original activities F1-scores

(g) Grouped activities Weighted F1-scores

(h) Original activities Weighted F1-scores

Figure 6.4 – Metrics comparison between baseline and previous activities models on Aruba, Milan and Cairo datasets

6.4 The Time of Activity Sequence Appearance

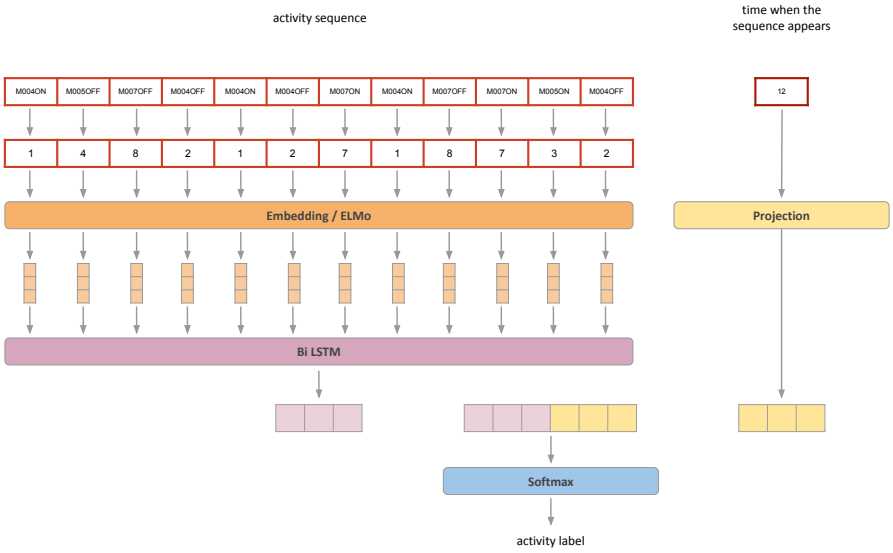
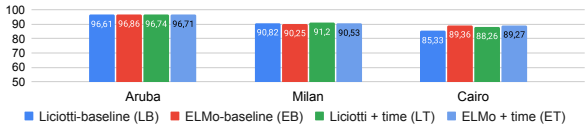


Figure 6.5 – Activity sequence and time input

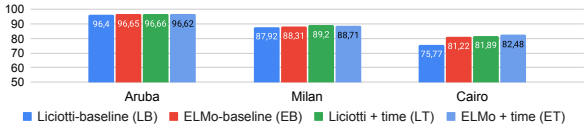
Many of our activities are related to a particular time in the day. Thus, it seems valuable to add this time information to recognize daily activities. For example, "Eating" occurs in general three times a day, in the morning ("Breakfast"), in the middle of the day ("Lunch") and in the evening ("Dinning").

We propose to add a second input that is the time value of when the activity sequence occurs, see Figure 6.5. The time value is the hour of the first element in the sequence. We chose to use the hour granularly because using minutes could be too precise as sensor activations don't usually appear at the exact same minute every days. This inclusion seems to allow the model to understand that particular times of the day are more conducive to the occurrence of given activities.

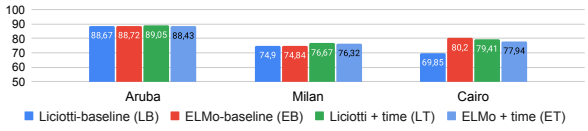
This second input is given to a projection layer, which is concatenated in the result of the bidirectional LSTM output, before the softmax layer. Such a configuration allows the softmax layer to use the activity sequence and the time representation to conclude the task classification. We note LT for Liciotti + *time* and ET for ELMo + *time*.



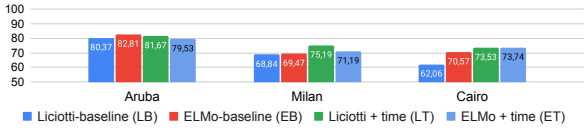
(a) Grouped activities Accuracy



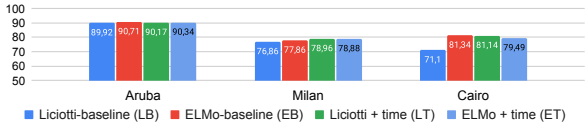
(b) Original activities Accuracy



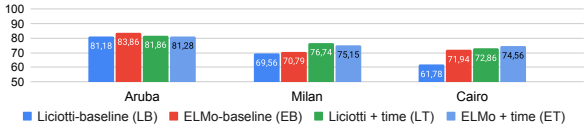
(c) Grouped activities Balanced Accuracy



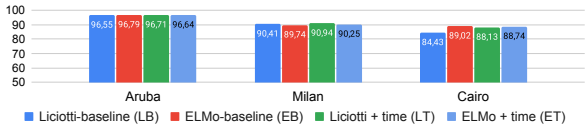
(d) Original activities Balanced Accuracy



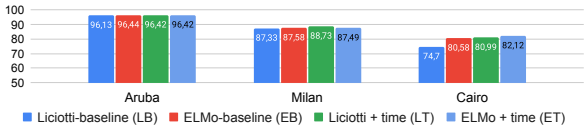
(e) Grouped activities F1-scores



(f) Original activities F1-scores



(g) Grouped activities Weighted F1-scores



(h) Original activities Weighted F1-scores

Figure 6.6 – Metrics comparison between baseline and time extension models on Aruba, Milan and Cairo datasets

In Figure 6.6 we can observe the comparison metrics results between baseline and time extension models. If we look at the accuracy, performance seems similar between baseline models and extended models.

The Balanced Accuracy reveals various improvements, in particular for the LT model which improves about 9.22% compared to the LB model on the original Milan dataset and 18.48 % for the dataset Cairo. The ET model did not improve the performance of the EB model as much and slightly worsened the performance on the Aruba dataset. The improvements to the ET model appears primarily on the dataset Cairo with the original labels (4.5%) and on the both versions of the Milan dataset. The F1-score metrics show how the time information improved baseline models, in particular for the Milan and Cairo datasets.

The time information seems to help the model to better recover class (F1-score) and to be more balanced between classes (Balanced Accuracy). Results show a high improvement of classification performance, in particular for the LT model. As for the ET model, it seems to perform better on the Cairo dataset.

6.5 The Time of Sensor Event Occurrence

As discussed above, time may be important for activity recognition. Here we propose to extend the previous structure by using a sequence of time values in addition to the sequence of sensor activations as input into the model, see Figure 6.7. Each time value corresponds to the time at which the sensor event occurred in the sensor sequence. This temporal representation should allow the model to understand how time changes over the course of the activity, which can help it encode the fact that certain activities occur at a particular time of day and have a certain duration. Although duration can be interpreted by the length of the sequence, i.e., the number of elements in the sequence, this temporal representation should help the model to understand that some activities only take place during a specific time of day. We note LTS for Liciotti + *time sequence* and ETS for ELMo + *time sequence*.

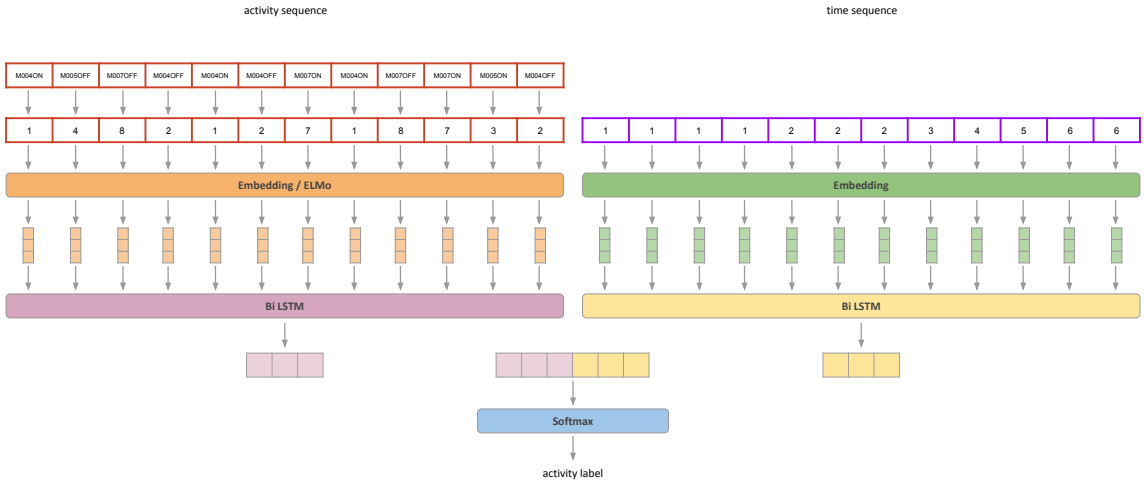
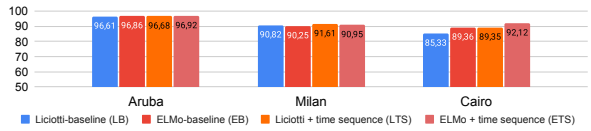


Figure 6.7 – Activity sequence and time sequence

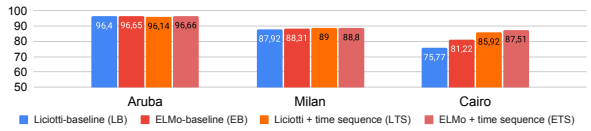
The Figure 6.8 presents the results between baseline and time sequence extension models. We can observe on it that the ETS model outperforms all other models on every dataset, but in particular shows the best improvements with the Milan and Cairo datasets. The Balanced Accuracy and the F1-score rise for both dataset versions, except for the Aruba dataset where performance is still almost the same. The LTS model performed better on the Milan and Cairo datasets than the LB model, but decreased in performance on the Aruba dataset. For both models the best improvements can be seen on the Cairo dataset. The LTS improves 15.7% for the grouped labels version and 28.58% for the original labels version in Balanced Accuracy. It rose also to 14% for the grouped labels version and 31.28% for the original labels version in F1-score.

In the meantime, the ETS model improved 3% and 17.27 % on the grouped and original dataset versions in Balanced Accuracy. In addition, this model increased the F1-score by 4.7% and 16.45% respectively for grouped and original dataset versions.

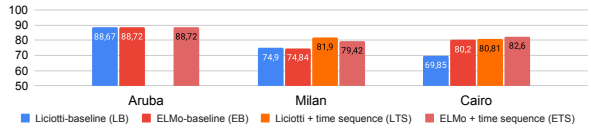
For both models the gap between the F1-scores and Weighted F1-sores, similar for Accuracy and Balanced Accuracy, becomes smaller. This means that despite the unbalanced factor of the dataset both models recognize all activities in a balanced way.



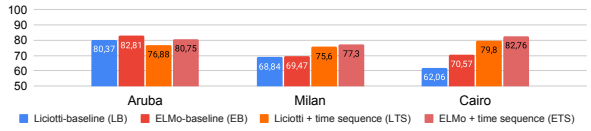
(a) Grouped activities Accuracy



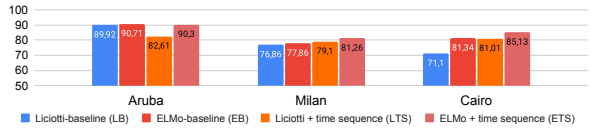
(b) Original activities Accuracy



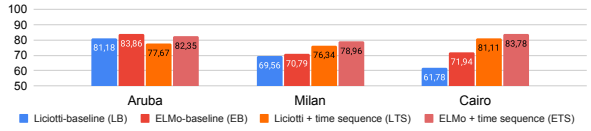
(c) Grouped activities Balanced Accuracy



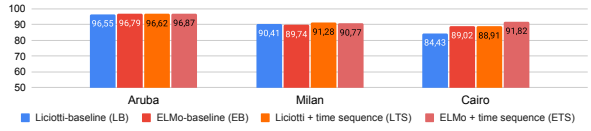
(d) Original activities Balanced Accuracy



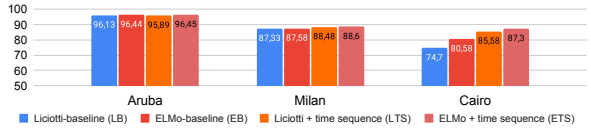
(e) Grouped activities F1-scores



(f) Original activities F1-scores



(g) Grouped activities Weighted F1-scores



(h) Original activities Weighted F1-scores

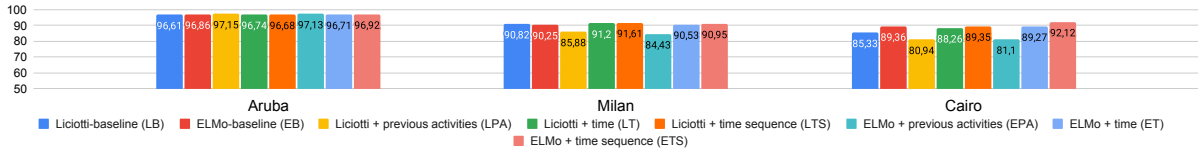
Figure 6.8 – Metrics comparison between baseline and time sequence models on Aruba, Milan and Cairo datasets

6.6 All Extension Comparisons

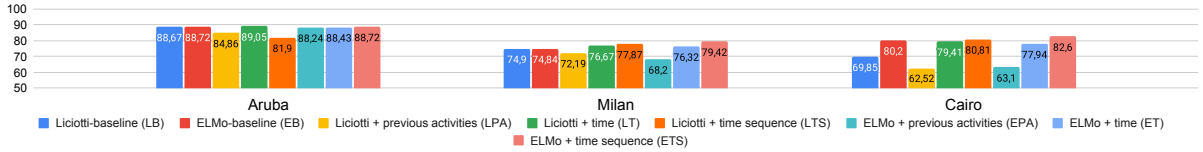
6.6.1 Grouped Activities: Global Analysis

We first studied the performance of the different models on the three re-labeled datasets. Figure 6.9 contains a comparison of the scores obtained by each model.

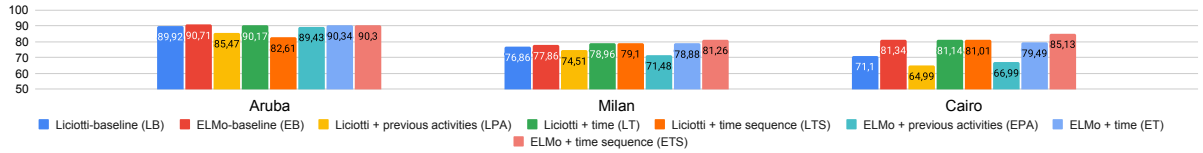
(a) Accuracy



(b) Balanced accuracy



(c) F1-score



(d) Weighted F1-score

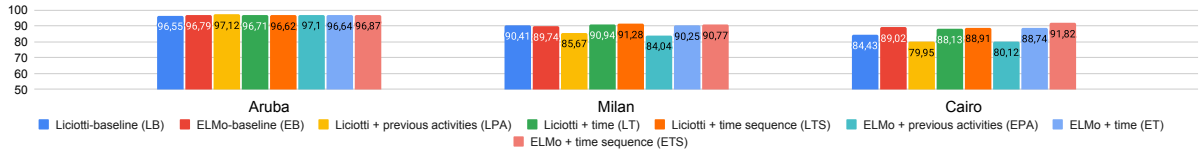


Figure 6.9 – Accuracy, Balanced Accuracy, F1-score and Weighted F1-score comparison on datasets relabeled activities

On the Aruba dataset, all the models had a similar accuracy and F1-score. A slight improvement can be noted for the LPA and EPA models in terms of accuracy. The LT model had the best balanced accuracy score, followed very closely by the ET sequence and EB models. If we look at the F1-score for this dataset, the EB model remains the best, followed by the ET and ETS models. This last metric allows us to conclude that the models using the ELMo embedding seem to be more balanced in terms of class recognition for this dataset. Indeed, a high F1-score means that recall and precision are also high.

On the Milan dataset, the LTS and LB models come second and third in terms of accuracy after the EB model, while the other ELMo based models are not far behind. In terms of balanced

accuracy and F1 score, the ETS model outperforms all other approaches. It seems that time is more important than previous activities' information.

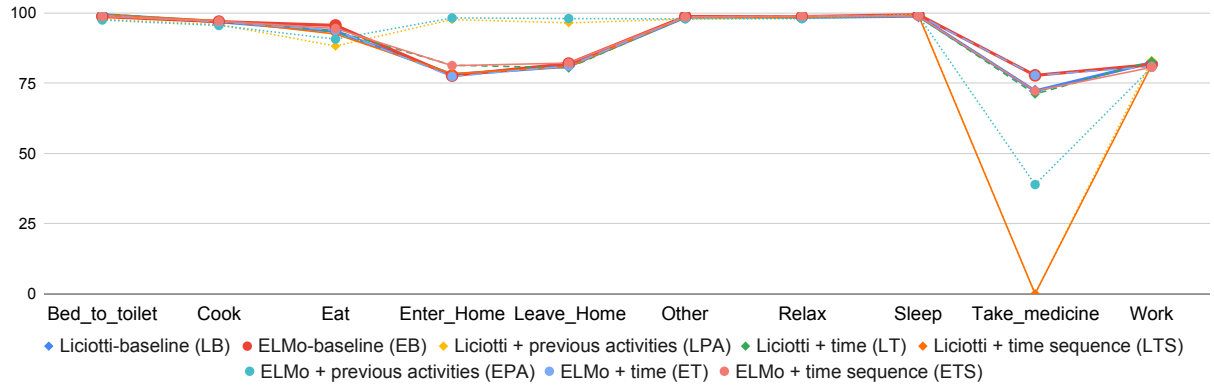
For the Cairo dataset, the EB model already produces very high performances with all metrics and far exceeds the performance of the LB model. However, we can observe that by adding temporal information (LT, LTS, ET, ETS), the models obtain important gains. In terms of accuracy and F1-score, this version shows that the LB model produces equal or even slightly lower scores than the EB model (less than 1%). The addition of time allows the LTS model to improve performance by about 15% in balanced accuracy and about 14% in F1-score. The ETS model again produces the best scores and increases balanced accuracy by 3% and F1-score by 4.6% compared to EB.

6.6.2 Grouped Activities: Class-by-Class Analysis

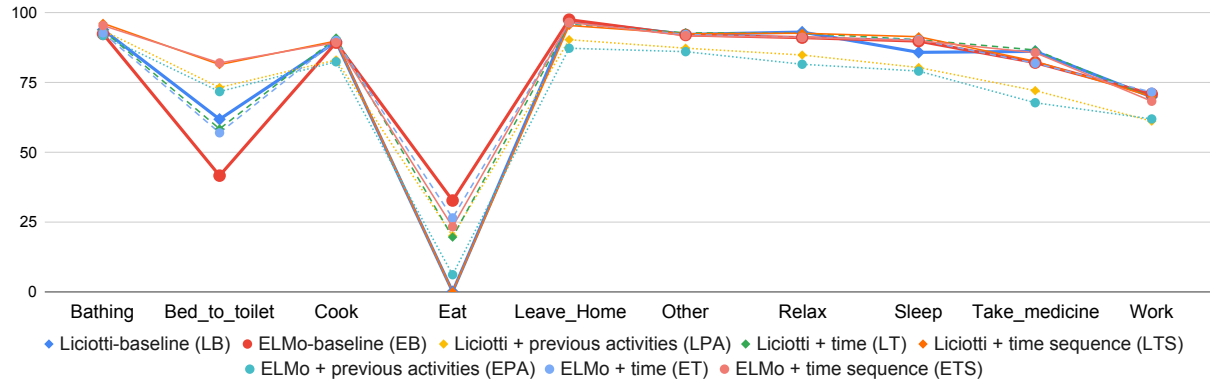
In order to deepen our analysis, we studied class-by-class the performance of each model via the F1-score metric. Figure 6.10 is a visual overview of how each model recognized each activity for the three grouped dataset versions, while Table 6.1 contains the detailed F1-score values. Thanks to these two representations, we can conduct the following analysis.

Table 6.1 – F1-score by models and by relabeled activities of each dataset

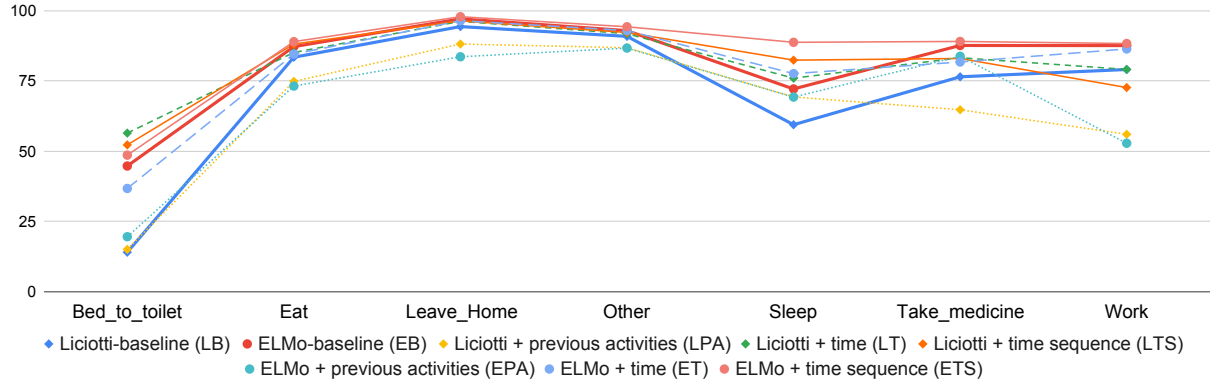
		Liciotti Baseline (LB)	ELMo Baseline (EB)	Liciotti + previous activities (LPA)	Liciotti + time (LT)	Liciotti + time sequence (LTS)	ELMo + previous activities (EPA)	ELMo + time (ET)	ELMo + time sequence (ETS)
Aruba	Bed_to_toilet	99,37	98,72	97,81	**99,69**	99,36	97,48	99,05	98,72
	Cook	96,89	96,92	95,89	96,88	97,29	95,53	96,84	**97,14**
	Eat	93,46	**95,58**	88,21	93,52	92,50	90,68	93,82	94,47
	Enter_Home	78,07	77,67	97,67	81,40	78,37	**98,23**	77,36	81,24
	Leave_Home	81,02	81,99	96,48	80,45	80,87	**98,01**	81,13	82,16
	Other	98,49	**98,74**	97,83	98,50	98,58	97,89	98,58	98,62
	Relax	98,45	98,72	98,24	98,66	98,61	97,96	98,84	**98,91**
	Sleep	98,88	99,38	**99,50**	98,74	99,13	98,99	99,12	98,85
	Take_medicine	72,23	**77,78**	0,00	71,12	0,00	38,89	**77,78**	72,23
	Work	82,34	81,62	**83,09**	82,79	81,39	80,65	80,91	80,63
Milan	Bathing	94,13	92,32	93,51	93,56	**96,10**	91,73	92,67	95,31
	Bed_to_toilet	61,94	41,55	73,12	58,30	81,32	71,65	56,89	**81,91**
	Cook	90,42	89,19	82,98	**90,85**	89,77	82,32	89,95	89,22
	Eat	15,39	**32,64**	19,97	19,53	0,00	6,07	26,41	23,24
	Leave_Home	96,53	**97,40**	90,24	95,79	95,32	87,17	96,29	96,46
	Other	**92,88**	91,93	87,20	92,69	92,35	85,95	92,09	91,68
	Relax	**93,20**	90,98	84,75	92,53	92,55	81,44	91,06	91,04
	Sleep	88,24	89,74	80,27	90,22	**91,30**	78,99	90,22	89,88
	Take_medicine	84,57	82,04	72,01	**86,55**	82,50	67,66	81,80	85,60
	Work	68,93	70,79	61,07	69,57	**69,83**	61,85	71,47	68,20
Cairo	Bed_to_toilet	14,02	44,71	15,02	**56,43**	52,23	19,53	36,71	48,58
	Eat	83,45	87,23	74,85	85,15	88,13	73,18	84,55	**89,00**
	Leave_Home	94,36	97,11	88,15	96,23	96,40	83,61	96,46	**97,88**
	Other	90,88	92,92	86,85	91,83	92,25	86,71	92,95	**94,30**
	Sleep	59,42	72,15	69,30	75,99	82,41	69,26	77,61	**88,76**
	Take_medicine	76,48	87,64	64,75	83,27	83,00	83,80	81,75	**89,07**
	Work	79,08	87,59	55,99	79,10	72,66	52,82	86,40	**88,33**



(a) Aruba



(b) Milan



(c) Cairo

Figure 6.10 – Algorithms performances for each grouped activity on Aruba, Milan and Cairo

On the Aruba dataset, at first glance of Figure 6.10, all the models seem to be similar to each other during each activity. The LPA and LTS models were not able to find the "Take medicine" class which only contained the "Respirate" activity and only occurred six times in the dataset. Nevertheless, the LPA and EPA models achieved a very large improvement on the "Enter Home" and "Leave Home" activities, about 20% compared to the baseline model for the EPA model. It seems that past activity information is important for recognizing these two activities.

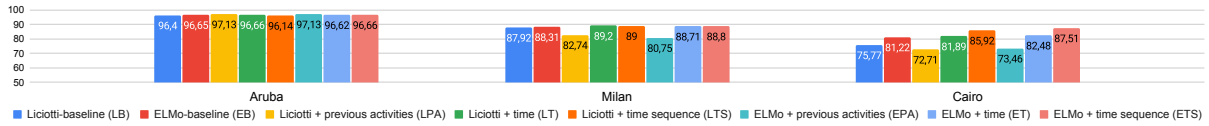
The Aruba dataset house contains several entries into and out of the house and it is not uncommon for the resident to exit by one entry and enter by another. Moreover, we notice that the resident sometimes makes several entries and exits in a short period of time. In this case, the model observes, thanks to the addition of the sequences of preceding activities, that one must leave before entering and vice versa. This helps the model to differentiate these two activities. However, even though the EPA model is close to the other models during the other tests, it loses about 50% of its performance on the "Take medicine" activity.

With the Milan dataset, all the models have similar performance in eight out of ten cases, and the LPA and EPA scores are always under the other models. We noticed that the most difficult activities to recognize were "Eat" and "Bed to toilet", all models performed poorly on these two classes. The model with the best results for the class "Eat" is the EB model. Against all expectations, it was not the models which used time that had the best performances for this activity, but the ET and ETS models which closely compared to Liciotti's-based models. This could be explained by the fact that the sensors' activation context is more important, since the sensor used to recognize this activity can be used when a resident is traveling due to their location. In other words, the PIR above the dining table may be activated when the resident passes by since the table is on the way between the kitchen and the living room. While the absence of information on time and previous activities seems to give better performances than the EB model, especially for the "Eat" activity, the models using the above-mentioned information give the best performances on the "Bed to toilet" activity. The LTS and ETS models increase the performance significantly for the "Bed to toilet" activity. The ETS model improves 97% over the EB model and the LTS model improves 31.28% over the LB model. In addition, it seems that using information from the previous activities also increases the performance compared to the baseline models, but less so than the time under sequence form.

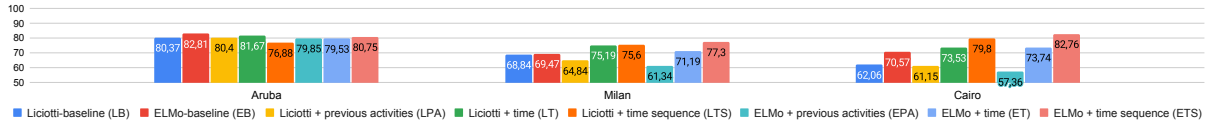
For the Cairo dataset, two activities seem difficult to recognize, "Bed to Toilet" and "Sleep". They are indeed very close to each other as these two activities occur during the same time of day (Night). In addition, as the dataset contains the activities of two resident, one resident can go to toilet while the second one is sleeping. The ETS model obtains the best performance for the activity "Sleep", it results in a significant increase in performance over the EB model (23%). For the "Bed to Toilet" activity, the ETS model is slightly better than the EB model, but the best model is the LT model.

In summary, we can see that using information from previous activities does not seem to improve the performance of the two baseline models, except for the recognition of the activity classes “Enter Home” and “Leave Home” in the Aruba dataset. The addition of time and its interpretation in sequential form seems to produce performance improvements for the dataset. This increment allows us to improve the performance of the algorithms for the Cairo dataset, in particular for the Liciotti model. In general, the ETS model produces the best performances for the three datasets. The addition, time sequence inclusion, combined with ELMo embedding, seems to be a powerful method for the activity sequence classification problem.

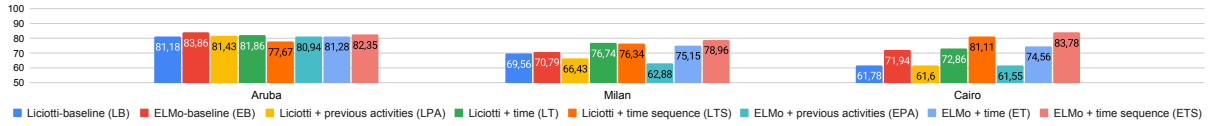
6.6.3 Activities of Origin: Global Analysis



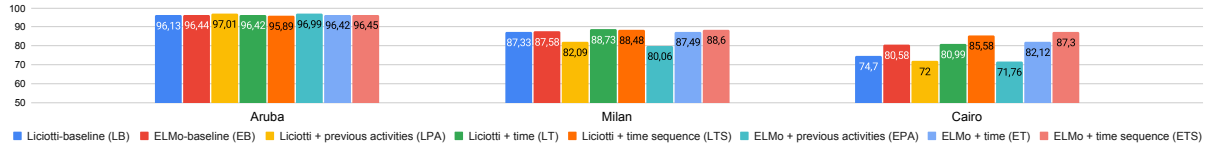
(a) Accuracy



(b) Balanced Accuracy



(c) F1-score



(d) Weighted F1-score

Figure 6.11 – Accuracy, Balanced Accuracy, F1-score and Weighted F1-score comparison on datasets activities of origin

Figure 6.11 presents a comparison of the different approaches for the datasets with their origin labels. They all obtain very similar results on the Aruba dataset. The accuracy score is equivalent to the score from the previous experiment. However, the balanced accuracy rating has decreased, which may be caused by the increase in the number of classes, with very similar classes, e.g. “Meal preparation” and “Wash dishes”. The Balanced Accuracy allows us to determine that these approaches can recognize a greater number of different classes of activities. The EB model

produces the best score for Balanced Accuracy (82.81%), followed by the LT model (81.67%) and ETS (80.75%). In terms of F1-score, the EB model shows, once again, the best performance for this dataset (83.86%) followed by the ETS (82.35%) and ET (82.35%) models. The improved Liciotti based models gain very little compared to the LB model. The LTS model drops in number of recognized classes, as well as balanced accuracy and F1-score. The high weighted F1-score metric for this approach allows us to conclude that for this dataset, the model seems to recognize the classes with the most occurrences.

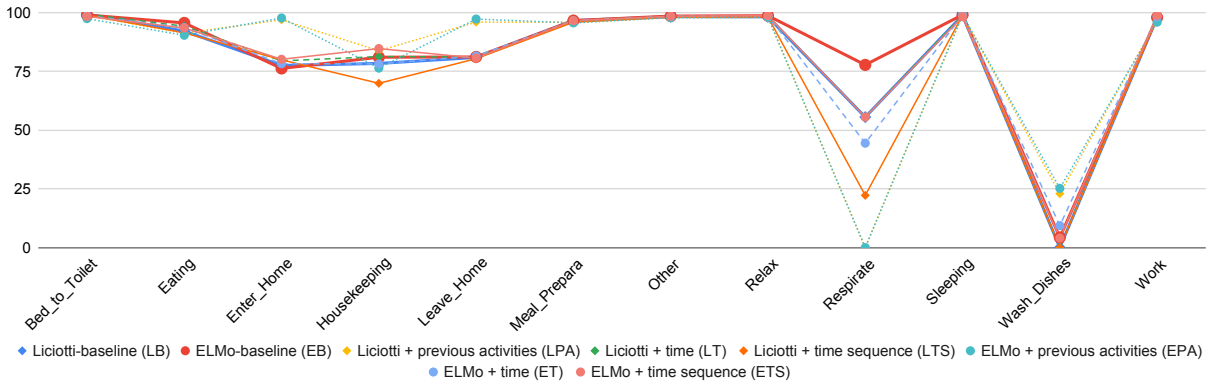
Concerning the Milan dataset, the LT, LTS, ET and ETS approaches improved performance (89.2%, 89%, 88.71%, 88.8%), compared to the respective baselines (LB: 87.92%, EB: 88.31%). The EB model already had a slight lead over the LB model, which was solidified by the Liciotti and ELMo extended approaches. Indeed, the ETS approach outperforms all the others in terms of balanced accuracy, F1-score and weighted F1-score. The latter seems to be the most efficient and balanced approach for this dataset. Nevertheless, the LT and LTS approaches are not outdone as they perform better than the LB, and are close to the performance of the ETS approach. Once again, the LPA and EPA approaches drop in performance, as with the previous experiment.

For the Cairo dataset, the EB approach performs better than the LB approach in accuracy (81.22% versus 75.77%), balanced accuracy (70.57% versus 62.06%) and in F1-score (71.94% versus 61.78%). This gap is reduced by the LT and LTS approaches, which catch up and largely exceed, in particular the LTS model, the performances of the EB approach. However, the ETS approach keeps a significant lead, especially in balanced accuracy and F1-score. The "previous activities" approaches do not improve the performances for this dataset.

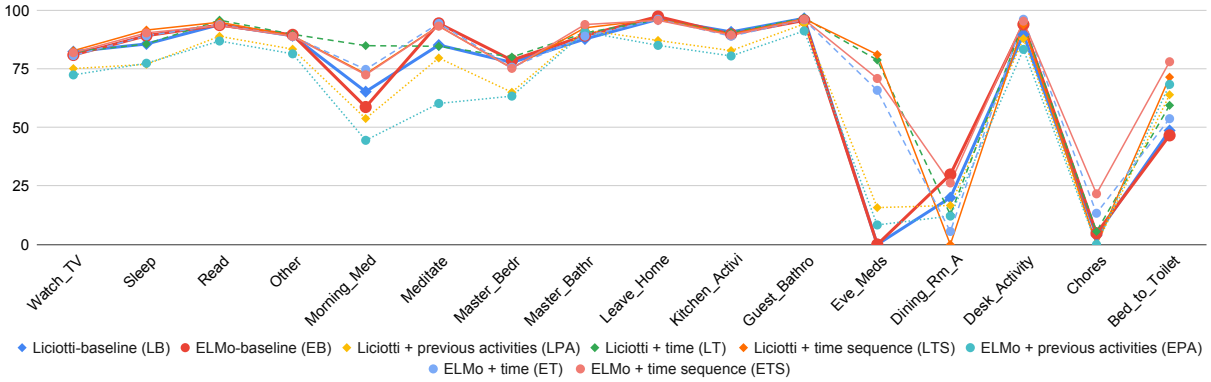
6.6.4 Activities of Origin: Class by Class Analysis

As before, we studied in more detail the performance of each model for each class of datasets (see Figure 6.12 and Table 6.2).

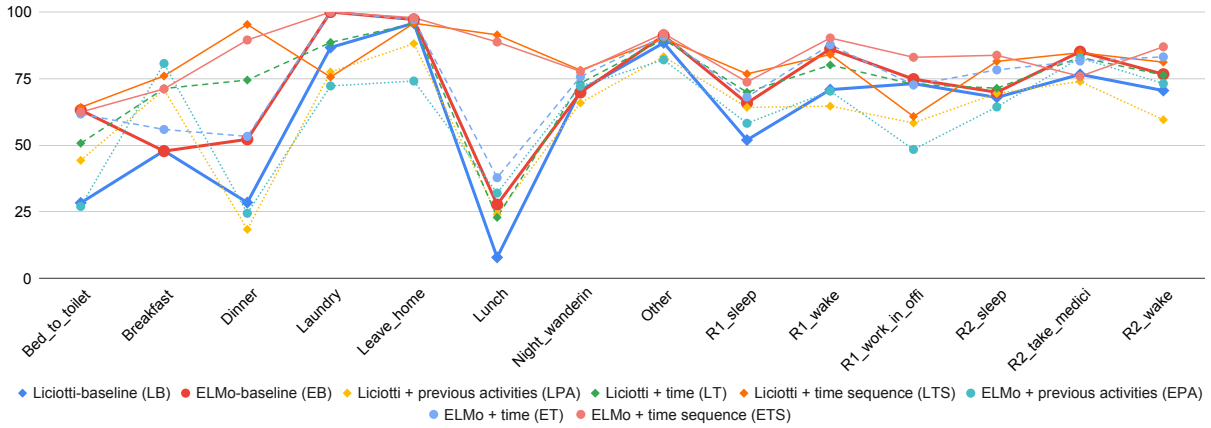
Concerning the Aruba dataset, the hardly recognizable classes are "Respirate" and "Wash dishes". "Respirate" corresponds to an activity using a medical breathing apparatus and appears only six times in the dataset. In terms of sensor activation, the activity "Wash dishes" is similar to "Meal preparation". The "Wash dishes" class of activity is poorly recognized or completely unrecognized by baseline models. Looking at the results of the approaches using time, it seems that the temporal information does not grant better recognition of this activity. However, the LPA and EPA models received the best scores for this activity with a F1-score around 25%, it seems that only these two approaches worked. For the activity "Respirate" the EB performed the best with a F1-score of 77.78%, followed by the ETS, LT and LB models. We noticed that the use of information from previous activities significantly increases the performance for recognizing the classes "Enter Home" and "Leave Home", as was the case previously. Globally the two best models on this dataset and configuration, labels of origin, are EB and ETS.



(a) Aruba



(b) Milan



(c) Cairo

Figure 6.12 – Algorithms performances for each activity of origin on Aruba, Milan and Cairo

Table 6.2 – F1-score by models and by activities of each dataset

		Liciotti Baseline (LB)	ELMo Baseline (EB)	Liciotti + previous activities (LPA)	Liciotti + time (LT)	Liciotti + time sequence (LTS)	ELMo + previous activities (EPA)	ELMo + time (ET)	ELMo + time sequence (ETS)
Aruba	Bed_to_Toilet	**99,37**	99,05	98,77	99,36	99,04	97,48	98,72	98,41
	Eating	92,19	**95,65**	91,30	94,26	91,43	90,39	93,68	93,51
	Enter_Home	77,53	76,27	96,97	79,49	79,94	**97,76**	78,29	80,18
	Housekeeping	78,40	81,05	83,81	81,33	69,94	76,41	78,20	**84,74**
	Leave_Home	80,97	81,17	96,01	81,15	80,43	**97,31**	81,56	80,70
	Meal_Preparation	**96,67**	96,65	95,93	96,59	95,94	95,66	96,61	96,64
	Other	98,33	**98,58**	97,93	98,54	98,06	97,88	98,48	98,49
	Relax	98,24	98,73	98,00	98,54	98,18	98,05	98,74	**98,76**
	Respirate	55,56	**77,78**	0,00	55,56	22,23	0,00	44,45	55,56
	Sleeping	98,88	**99,12**	99,00	99,00	98,88	99,11	98,99	98,34
	Wash_Dishes	0,00	4,31	22,95	0,00	0,00	**25,23**	9,37	4,05
	Work	98,02	98,00	96,54	98,54	98,00	95,98	98,30	**98,85**
Milan	Bed_to_Toilet	48,68	46,60	63,96	59,44	71,47	68,37	53,74	**78,05**
	Chores	3,93	4,77	0,00	5,56	0,00	0,00	13,34	**21,64**
	Desk_Activity	89,73	93,99	87,89	94,53	93,76	83,33	**96,19**	95,39
	Dining_Rm_Activity	20,00	**29,90**	16,67	12,73	0,00	12,13	5,56	26,28
	Eve_Meds	0,00	0,00	15,75	78,79	**81,12**	8,34	65,86	70,91
	Guest_Bathroom	96,60	95,72	94,19	**96,76**	96,43	91,24	95,99	96,14
	Kitchen_Activity	**90,79**	89,79	82,80	90,52	89,79	80,52	88,98	89,46
	Leave_Home	96,19	**97,39**	87,16	96,03	96,24	85,05	96,21	95,76
	Master_Bathroom	87,77	89,25	91,58	90,36	92,55	90,75	89,00	**93,95**
	Master_Bedroom_Activity	77,73	78,72	65,03	**80,07**	75,35	63,37	75,72	75,24
	Meditate	85,19	**94,45**	79,66	84,65	93,34	60,24	**94,45**	93,27
	Morning_Meds	65,28	58,72	53,80	**84,93**	72,90	44,49	74,78	72,47
	Other	89,14	89,55	83,43	89,79	89,14	81,44	88,85	**88,92**
	Read	93,88	93,72	88,92	**95,82**	94,94	86,91	93,88	93,52
	Sleep	85,64	89,18	76,94	85,62	**91,59**	77,42	89,14	90,37
	Watch_TV	**82,38**	80,94	75,14	82,28	82,77	72,43	80,67	81,93
Cairo	Bed_to_toilet	28,32	63,16	44,28	50,71	**64,20**	26,99	61,68	62,43
	Breakfast	47,90	47,76	70,91	71,27	75,99	**80,67**	55,89	71,12
	Dinner	28,44	52,18	18,37	74,45	**95,32**	24,42	53,30	89,50
	Laundry	86,67	**100,00**	77,47	88,58	75,56	72,23	**100,00**	**100,00**
	Leave_home	95,83	97,17	88,14	95,68	95,78	74,09	97,17	**97,88**
	Lunch	7,85	27,65	24,04	22,86	**91,42**	32,01	37,79	88,75
	Night_wandering	70,54	69,79	65,88	73,24	**78,16**	72,14	75,66	77,73
	Other	88,42	91,15	83,24	89,75	90,13	82,05	90,88	**91,95**
	R1_sleep	51,92	65,78	64,18	69,85	**76,76**	58,21	68,06	73,71
	R1_wake	70,86	86,11	64,65	80,10	84,05	70,31	87,72	**90,25**
	R1_work_in_office	73,11	74,79	58,32	72,86	60,77	48,39	72,63	**83,00**
	R2_sleep	67,92	69,86	69,45	71,39	81,47	64,34	78,27	**83,76**
	R2_take_medicine	76,59	**85,10**	73,91	82,98	84,72	82,78	81,60	75,86
	R2_wake	70,50	76,64	59,53	76,30	81,16	73,13	83,16	**86,93**

In the original Milan dataset some activities, such as “Eve Meds”, “Morning meds”, “Dining RM Activity”, suggest by their denomination that it is necessary to consider temporal information. Indeed, we noticed that the best scores for this dataset were obtained by methods using time. While the baseline models were unable to recognize the “Eve Meds” activity, the LT, LTS, ET and ETS approaches obtained an F1-score higher than 65%. The LTS approach obtained the best score with 81.12%. The “Morning Meds” activity which was previously recognized with a F1-score between 58% and 65% on the baseline models obtained an F1-score between 72% and 84% with the models integrating time.

Moreover, we noticed that the ETS model could improve the F1-score of 64.60% with the EB model to 78.05%. This same model managed to have the highest F1-score of 21.64% on the “Chores” activity. The “Dining Rm Activity” was better recognized by the EB and ETS approaches. The ETS model appears to get the best scores globally.

The Cairo dataset also offers a number of time-dependent activities, “Breakfast”, “Dinner”, “Lunch”, “Night Wandering”, etc. Although the EB model already performs well on these activity classes, it seems that the use of time improves the recognition scores of these classes. The ETS and LTS models give the best results for all activities with high F1-score. For the activity “Lunch” the best baseline model is EB with a score of 27.64%, while the LTS approach obtained the overall best score of 91.42% and the ETS got 88.75%, which correspond to an increase of times 3.3 and times 3.2, respectively. We notice that in this dataset the ETS method obtained the best score for 50% of the classes and the second-best score for the remaining 50%.

6.7 Summary

In this chapter we have proposed to extend our model, the bidirectional LSTM with ELMo pretrained embedding, and the Liciotti et al. [3]'s model, the bidirectional LSTM with embedding, using time information or previous activities' context. The results presented in this chapter show that the proposed approaches lead to a viable solution that significantly improves the task of recognizing ADLs in a smart home scenario. After a full comparison of the two basic models, we find that the extended models, applied to three CASAS datasets, clearly perform better. This improvement is particularly strong for the ELMo model combined with sequential time information.

We have mainly explored two enhancement strategies, one that uses a larger contextual sequence as input and one that uses temporal information. For the temporal approach we first used the time of occurrence of the activity sequence, i.e. when the activity occurs, and then a sequence of temporal events, i.e a sequence containing the time of occurrence of each event in the sensor activation sequence. We have observed that the contribution of time allows a better classification rate, but also a better equilibrium in terms of number of recognized classes, which is visible thanks to the Balanced Accuracy and F1-score metrics.

However, while the addition of temporal information seems to improve the results, the use of a larger context does not. On the contrary, the use of more contextual information seems to decrease the classification performance. We suggest that this loss of performance occurs because sequences become more similar to each other with the addition of previous activities.

Another hypothesis could be the sequence length, as the input models can become too large enough for the LSTM structure. The average sequence length for the Cairo dataset, for example, increases from $\sim$ 600 to $\sim$ 1800 elements, which suggests to us the possibility of a potential vanishing gradient problem [2]. In a future work, it could be interesting to investigate the proposed approach by using Transformer Structures [6] in order to avoid such a limitation.

Moreover, originally, activity sequences had different lengths and activities belonging to the same class had a similar length. By adding previous activities, we destroy this potentially

important knowledge that seems useful for the LSTM structure. Nevertheless, we can observe exceptions in the recognition of activities such as “Leaving Home”, “Enter Home” and “Wash Dishes” on the Aruba dataset. The use of a wider context allows recognizing an activity that was not recognized before, “Wash Dishes”, but also improves the F1-score by about 20% for the activities “Leaving Home” and “Enter Home”. It appears that previous activities’ knowledge seems to help in some cases.

In conclusion, we have seen that the approach combining ELMo and time sequence obtains the best performance results in general on every dataset. This revelation seems to confirm the effectiveness of using an ELMo pretrained embedding. We achieved the highest improvement with the Cairo dataset, which is the most challenging dataset in this study, as the activities of two residents were monitored. The Licotti’s model significantly improved in all test cases by using the time sequence and could even challenge more ELMo based models.

However, trying to imagine and create the best possible approach to perform the complex task of human activity recognition in a smart home environment is not enough. It is necessary to be able to apply these methods in a real-world setting. This task is very difficult to implement since these models and methods need a large amount of labeled training data from the real final system. This is difficult to obtain, as it is hardly possible to ask an end user to label this data over a long period of time. We will see in the next chapter a proposal to consider the application of a human activity recognition method in real homes.

Bibliography

[1] Maxime Devanne, Panagiotis Papadakis, and Sao Mai Nguyen. Recognition of activities of daily living via hierarchical long-short term memory networks. In *International Conference on Systems, Man and Cybernetics (SMC)*, pages 3318–3324. IEEE, July 2019.

[2] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[3] Daniele Liciotti, Michele Bernardini, Luca Romeo, and Emanuele Frontoni. A sequential deep learning application for recognising human activities in smart homes. *Neurocomputing*, 04 2019. doi: 10.1016/j.neucom.2018.10.104.

[4] Raúl Gómez Ramos, Jaime Duque Domingo, Eduardo Zalama, and Jaime Gómez-García-Bermejo. Daily human activity recognition using non-intrusive sensors. *Sensors*, 21(16): 5270, 2021.

[5] Tan-Hsu Tan, Munkhjargal Gochoo, Shih-Chia Huang, Yi-Hung Liu, Shing-Hong Liu, and Yun-Fa Huang. Multi-resident activity recognition in a smart home using rgb activity image and dcnn. *IEEE Sensors Journal*, 18(23):9718–9727, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Chapter 7

Generating Data for Human Activity Recognition in Real-World Applications

7.1 Introduction

In the previous chapters, we have seen human activity recognition approaches. We have proposed some improvements that constitute steps towards the deployment of real-world applications. Nevertheless, there is a gap between the application of HAR algorithms on the existing datasets and their deployment on real-world data.

Over the past few decades, the number of real smart home projects and test beds has been growing around the world. These multiple real-world environments produce datasets over periods ranging from days, weeks, to months with the help of volunteer residents. Some of these datasets are made public and allow evaluation of ADLs recognition algorithms and others. However, these datasets are not sufficient to cover the distribution of possibilities in terms of activities, lifestyles or house configurations. Indeed, they are limited in the number of activities, occupants they take into account, as well as in the types of sensors and residents' category. Therefore, algorithms trained on data from one particular environment will be difficult or impossible to reuse in others as they are adapted to a particular configuration of house, inhabitants and life habits.

Collecting new datasets to increase the number of examples is a very costly task in terms of equipment and difficult to accomplish. Even if fully equipped and ready-to-use environments, such as Amiqual4Home [7], can be used on an ad hoc basis, recruiting volunteers and collecting data is very time-consuming [4]. It is necessary to ask residents to live in the environment and to label their activities over several weeks or months, i.e., to track the start and end times of activities and the label of the activity. This involves a significant investment on the part of volunteers and presents a high risk of error in labeling. Moreover, there are very few such environments accessible, again limiting the diversity of environments. To overcome this, new data on households can be collected from customers by companies offering smart home solutions. However, this solution poses problems in terms of sufficient sensor equipment, anonymization,

privacy, security and regulation. Retraining a deep learning model on these new data should require to collect and annotate a large amount of data from the test case, which can be costly in terms of means and time.

The use of transfer learning techniques seems to be a solution to reducing the required amount of data in the test case. We have seen in the previous chapters that the use of pre-trained embeddings could allow this. Nevertheless, it is necessary to have a large amount of data for pre-training but also labeled data of the target environment to adjust the algorithms so that they are perfectly adapted. Assuming we have access to a large amount of data for the pre-training, from the point of view of a commercial real-world application, it is not possible to ask end-users or customers to label their activities themselves for several days, weeks or months. A digital copy of a target smart home, that having the same virtual sensors, could be a way to generate data to refine the algorithms for the physical target smart home.

The concept of Digital Twin consists in creating a complete virtual representation of a physical object, process or system. To address the challenges of data scarcity and, in particular, the gap between training data and use case data, data generation through the concept of Digital Twin seems to be a viable solution. In this chapter, we propose to bridge the gap between test and training data using a smart home simulator based on the Digital Twin concept. In the next section, we briefly explain the Digital Twin concept and its potential application for the ADLs recognition problem. In the third section, we will present the Virtual Home simulator and the proposed extension, Virtual Smart Home simulator, for the generation of Digital Twin smart homes. Then we will detail our process to generate valuable data from a Digital Twin smart home and how we evaluated them. Finally, we will evaluate an algorithm of ADLs recognition trained through this Digital Twin smart homes concept.

7.2 Digital Twin Smart Homes for Human Activity Recognition: Background and Concept

The Digital Twin concept is a particular simulation method. The objective of a Digital Twin is to be a complete virtual representation of a physical object, process or system. The digital representation provides both the elements and the dynamics of the operation of a device throughout its life cycle. Digital Twin models are gaining more and more interest for their potential and strong impact in application fields such as manufacturing, aerospace, healthcare, and medicine [2]. Nasaruddin et al. [8] defines the Digital Twin of a building as the interaction between the interior environment of a real building and a digital but realistic virtual representation model of the building environment. The latter enables real-time monitoring and data acquisition. For instance, a Digital Twin of a building have been used in [6] to find the strategic locations of sensors to collect data efficiently.

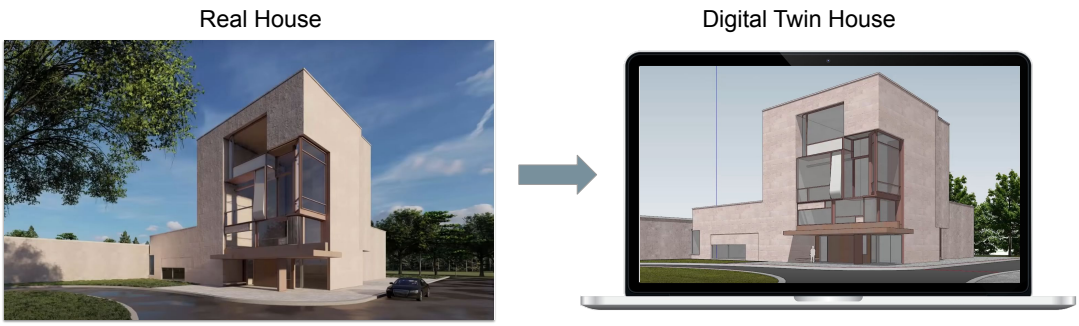


Figure 7.1 – Digital Twin of Smart Home Principe

In the context of ADLs recognition, a Digital Twin could be a digital copy of a target smart house equipped with the same virtual sensors (see Figure 7.1). In this digital environment, one or more avatars could perform ADLs by modeling complex resident behaviors. Thus, the virtual sensor logs could then be used to refine the final algorithms before deployment in the target house.

Besides, the Digital Twin concept could also be used to generate data from a large amount of houses and household configurations and residents' habits, with the possibility of accelerating the simulation, of automatic labelling, and without the cost of the sensors. For example, common data from a large number of Digital Twins homes could be used to generate a richer dataset. This large dataset could then be used for algorithm pre-training and finally refined on a target Digital Twin. Furthermore, a Digital Twin home may be used to evaluate the correct positioning and choice of sensors to recognize a predefined list of activities.

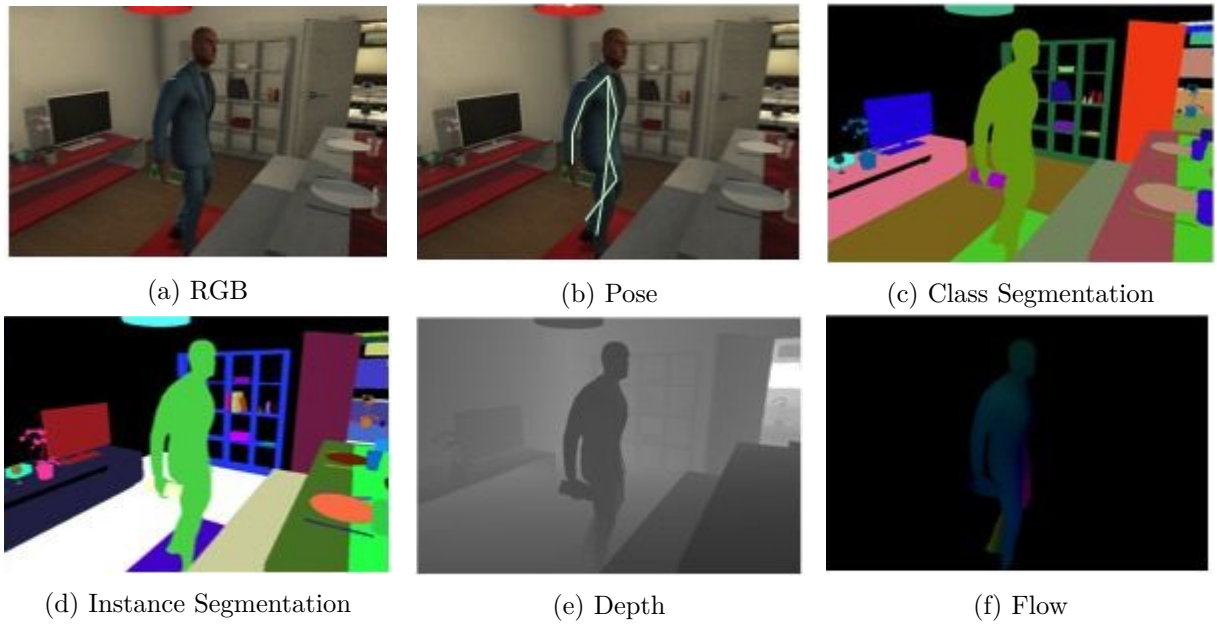
7.3 Virtual Smart Home: a Digital Twin Simulator

In order to develop this concept of Digital Twin for the smart home it is necessary to have a simulation environment. As we have seen in Section 2.8.2, the literature provides us with a number of smart home simulators with their advantages but also their disadvantages. The main problem of these simulation environments is their complexity of implementation but also their maintenance. These simulation environments often require 3D modeling and programming skills and the projects are often abandoned or not maintained. This makes it difficult to select a viable simulation environment for a Digital Twin approach.

However, in order to create a Digital Twin of a smart home, we have developed the Virtual Smart Home simulator. It is an extension of the Virtual Home simulator developed by Puig et al. [9]. The choice to use and extend this simulator was motivated by the fact that: 1) it is fully open source, therefore editable and extensible; 2) under development and has an active community; 3) it is designed and implemented to simulate human activities in a house.

7.3.1 Virtual Home

Virtual Home is a multiagent platform to simulate activities in a household. Agents are represented as humanoid avatars, which can interact with the environment through high-level instructions. These interactions can be chained under scripts to perform complex tasks and activities with multiple agents. Moreover, it includes a Knowledge Base, providing instructions to perform a large set of activities. It also can render videos of human activities, or train agents to perform complex tasks. It allows generating a large-scale video dataset of complex activities that are rich and diverse by simply recording the agent executing programs. Recordings provide dense ground-truth information, e.g. RGB, pose, segmentation, etc. (see Figure 7.2).



(a) RGB (b) Pose (c) Class Segmentation

(d) Instance Segmentation (e) Depth (f) Flow

Figure 7.2 – Virtual Home cameras types

Virtual Home is developed under the widely used Unity3D engine, a 3D game engine that is used in many commercial projects. This engine allows kinematic, physics and navigation models. It has a large developers' community, public library and stores of 3D models of objects, avatars and textures. It proposes seven furnished homes and 6 humanoid models. On average, each home contains 357 object instances (86 per room). To ensure visual diversity, it has at least 3 different models per class.

The different dwellings proposed are composed of rooms that can be reused to extend or create new configurations of homes. An apartment and agent example are shown in Figure 7.3a and Figure 7.3b. Furthermore, Virtual Home provides a Python API to interact with, and allow access to an environment graph that describes all object relations, hierarchical order and their status. This environment graph is updated when changes occur, which allow us to track them.

(a) Example of one of the seven apartments

(b) Example an avatar

Figure 7.3 – Apartment and avatar example

7.3.2 Virtual Smart Home

In summary, Virtual Home, enables the creation of scenarios where 3D avatars carry out activities of daily life and can be filmed via a virtual camera. Moreover, a database of several scripted activities is provided and can be applied on the avatars. The Virtual Home environment allows us to add and create new complete or piece of house via the Unity3D editor environment. Therefore, it allows us to create the digital reproduction of a home. Unfortunately, several features are missing to reproduce a smart house and generate sensor data. To be able to produce scenarios and collect sensor data, we have implemented the following new features into Virtual Home:

Virtual Sensors

We have implemented the possibility to log sensors and the five following virtual sensors: (1) opening/closing sensors; (2) pressure sensors; (3) lights (4); power consumption and (5) a floor presence sensor.

All sensors except the sensitive floor sensor were simulated using the environment graph of the scene. This graph lists all the objects in the scene with an ID and associates them with the states corresponding to their nature (e.g. closed/open, on/off). Power consumption sensor translates if an appliance is turned “on” or “off”.

The floor presence sensor is simulated using a downwards raycast from the avatar that checks the collision with the floor. As the floor of the digital house is divided by rooms, the intersection of the raycast with the floor rooms allows us to identify the room in which the avatar is located. The floor presence sensor does not directly give which avatar is in which room, it just gives the information of presence. In other words, it outputs if someone is or not in a room.

Interactive Objects and Avatar Actions

Some animations and interactions are not possible. Virtual Home proposes a large number of objects that can be included in the apartments, but many are passive and can not be interacted with by the agent Thus, we added new interactive objects to increase the number of possible interactions. We have added the possibility to open and close a drawer, turn on and off faucets on some object, open and close door of cabinets, etc. Moreover, the avatar can't perform some actions or activities that would be interesting for some ADLs. Therefore, we have implemented new avatar actions and animation such as "Lying" or "Eat".

Simulation Acceleration

Virtual Home not provide the possibility to change the speed of time flow. Thus, we have added the possibility to accelerate the time of the simulation. However, it is not possible to speed up time faster than the Unity game engine. Indeed, as Unity3D is a real-time render engine, the time factor cannot exceed the time required by the 3D engine that generates the rendering of the 3D scene. It takes a minimal amount of time for this engine to calculate the movements, interactions, and changes in the scene in addition to the visual rendering. We were able to accelerate the simulation time with a factor 4.

Configuration User Interface

In order to configure the simulation and start the data generation, we designed a graphical user interface (see Annex E). This interface is build to follow a three-step workflow. A first window allows the creation of an experiment configuration file specifying: (1) the simulation house; and (2) the set of sensors that we wish to monitor and log. A second window allows the creation of a scenario configuration file: (1) the start date of the experiment log; (2) the acceleration factor of the simulation; and (3) the different activities to play and their duration. The last window interface allows checking the configuration files, and to start the simulation.

7.4 The Smart Home Digital Twin

We conducted our experiments via the Experiment'Haal smart apartment, the IMT's living laboratory for health and autonomy [1], which aims to develop and host experimental assistive devices for user testing. This smart apartment consists of a main living space, a kitchen, a sleeping area and a bathroom (see Figure 7.4a). It allows simulating a domestic environment in order to validate the technological solutions proposed by the end-users in their daily life context, before they can be deployed on a larger scale.

We reproduced the apartment into the Virtual Smart Home Simulator to create its Digital Twin, (see Figure 7.4). All the elements of the apartment were measured and placed to be as close as possible to reality. However, due to the lack of a 3D model for some objects, we had to replace them with the most suitable model possible. We did not notice any difference or particular impact due to these adjustments at the time of the simulations.

(a) View of the living-lab [1] from a fish-eye camera (b) Virtual Smart Home simulator copy

Figure 7.4 – The real apartment and the Digital Twin

The apartment is equipped with cameras to observe experimentation's and various home automation sensors (see Figure 7.5) The apartment is also equipped with a connected floor that can track movements, trajectory and detect possible falls. We have defined zones corresponding to the rooms of the apartment in the connected floor software (see Figure 7.6). The connected floor log in which room the resident is and the transition between rooms.

The majority of sensors' apartments were reproduced to generate activities logs into the Digital Twin. Only motion, temperature and humidity sensors were not copied. The motion sensors were avoided as the apartment does not have walls, which allow a motion sensor to be

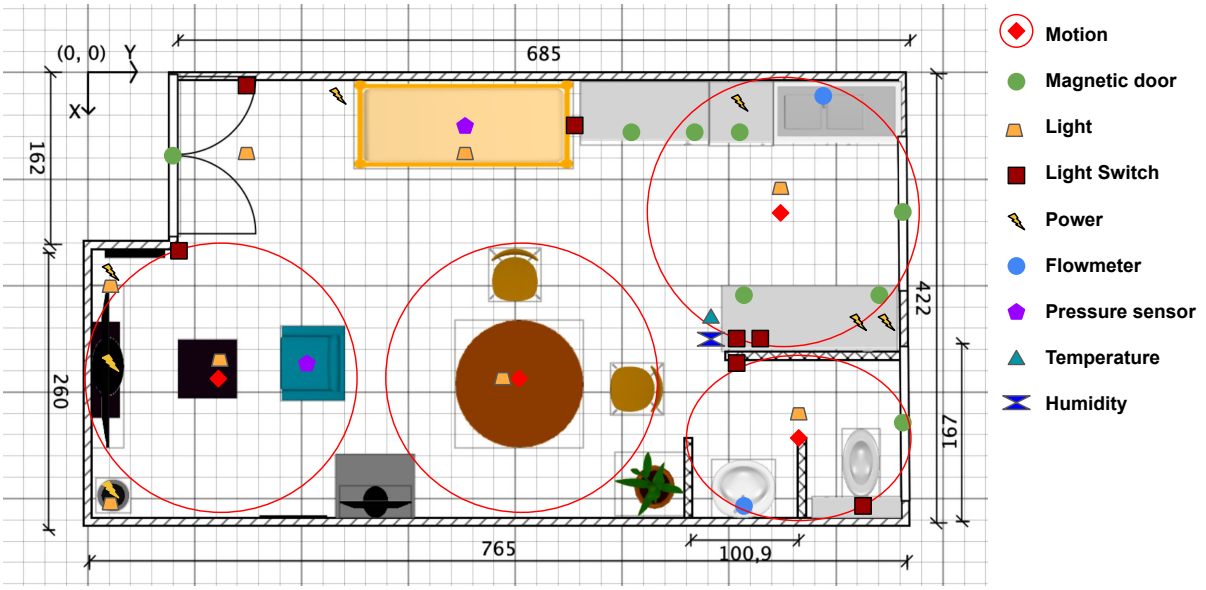


Figure 7.5 – Sensors locations

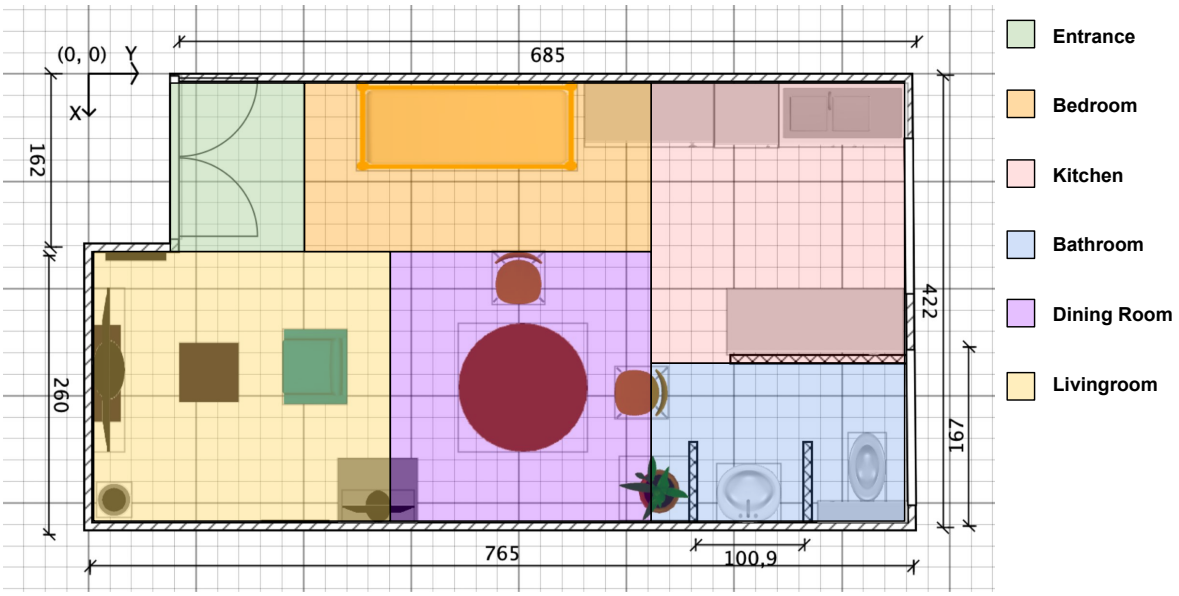


Figure 7.6 – Connected floor zones

triggered even if the resident is not in the room delimited for the sensor. In addition, the motion sensors used have a very high refresh time (120s), which limits the responsiveness and therefore the capture of transitions between rooms and presence. Temperatures and Humidity sensors were also not reproduced into the Digital Twin as we did not implement these sensors into the Virtual Smart Home simulator.

7.5 Digital Twin Data for Human Activity Recognition in Smart Homes

7.5.1 Data Generation Methodology

Our goal is to generate data from a Digital Twin apartment in order to train an algorithm, and then to evaluate the performance of the trained algorithm on real data from the apartment. To do this, we defined three scenarios that we called: “Morning”, “Middle of the day” and “Evening”. The morning scenario consists of simulating the activities from the time a resident wakes up until he or she leaves the apartment to go to work. The midday scenario is a simulation of the resident returning home, having lunch, and resting before returning to work. The last scenario, evening, represents the activities from the time the resident returns home until he goes to bed.

For these scenarios we have defined a list of possible activities labels : “Bathe”, “Cook breakfast”, “Cook dinner”, “Cook lunch”, “Dress”, “Eat breakfast”, “Eat dinner”, “Eat lunch”, “Enter home”, “Go to toilet”, “Leave home”, “Read”, “Sleep”, “Wash dishes”, “Watch TV”. These labels were inspired from labels of literature dataset such as CASAS [3], but principally by the ADLs definitions define by Katz [5]

We recorded real-world data from ten volunteers who played all the three scenarios in the smart apartment. Each volunteer performed each scenario at least once. Some volunteers performed the same scenario several times in different ways to add variability. The volunteers were recruited without any particular selection criteria. All of them were in good health and did not present any particular disorder because we did not aim at a medical framework. The main objective was to collect activity data without any particular anomalies to more easily reproduce them in the simulation environment.

Using a camera, we annotated the different activities as the volunteers performed them and describes them aloud. We asked the volunteers to express, just before the end of the current activity, the next activity that they perform. A new activity always starts at the end of the current activity. In this way, all sensor activations have an activity label.

Once all scenarios were recorded, we replicated the scenarios performed by the volunteers in the digital twin of the apartment. To do this, we used the video recordings of each volunteer’s scenarios to define activity scripts that exactly copy the volunteers’ actions. Each script contained the list and order of actions that the avatar had to perform to replicate the activities performed in the different scenarios. The avatar then executes the different scripts in the digital twin to generate the virtual sensor logs.

7.5.2 Synthetic and Real Data Post-Processing and Comparison

Unfortunately, because the transcription of the videos into a script is time consuming, we were only able to reproduce the scenarios of 3 volunteers in the Digital Twin which resulting in 15 reproduced scenarios over 62. We will compare and use the data of these 3 volunteers in the following, details in the Table 7.1.

Table 7.1 – Data scenarios details

Scenario Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Scenario Type	Morning	Morning	Evening	Mid-Day	Mid-Day	Evening	Mid-Day	Morning	Morning	Evening	Evening	Evening	Morning	Evening	Mid-Day
Subject	S9	S9	S9	S9	S9	S9	S7	S7	S7	S7	S7	S3	S3	S3	S3

Post-Processing

Before using the generated data for the algorithm training, we cleaned, analysed and compared the two datasets. Since the synthetic dataset only contains binary sensor activations and the smart apartment contains more sensors than the digital twin, due to the lack of implementation of more complex sensors such as the CO2 sensor, we had to remove all sensors activations that we haven't implemented. In addition, as smart outlets were used to monitor some appliance such as the television for example, we turn power consumption's into binary states. If the consumption of the television exceeds a certain threshold, then it is considered turned on, and vice versa.

For the synthetic data, we have removed the bad room transition. Indeed, we did not control the path finding algorithm of the avatar in the simulator. Thus, the avatar, in some cases, crosses multiple rooms before it reaches its room destination. For example, in the smart apartment, volunteers that start in the entrance and want to reach the kitchen cross rooms in this order: entrance, living-room, dining-room and then, kitchen. In some cases the avatar adds the cross of the bedroom before or after the dining-room. These particular cases could disturb the algorithm of ADLs recognition as during the recording, the volunteers have respected some passage points between rooms to simulate doors.

Basic Comparison

A first comparison consists in looking at the number of sensor activations in synthetic and real scenarios' data (see Figure 7.7). We can see that we have more sensor activations into real data than into synthetic data. This is explained as synthetic data doesn't have trouble and are "perfect" compared to real data. Indeed, the connected floor sensor sometimes loses and recovers the position of the resident, which increases the number of activations. Another example is the armchair pressure sensor, which alternates between "pressure detected" and "not pressure detect" in function of the position of the resident in the armchair.

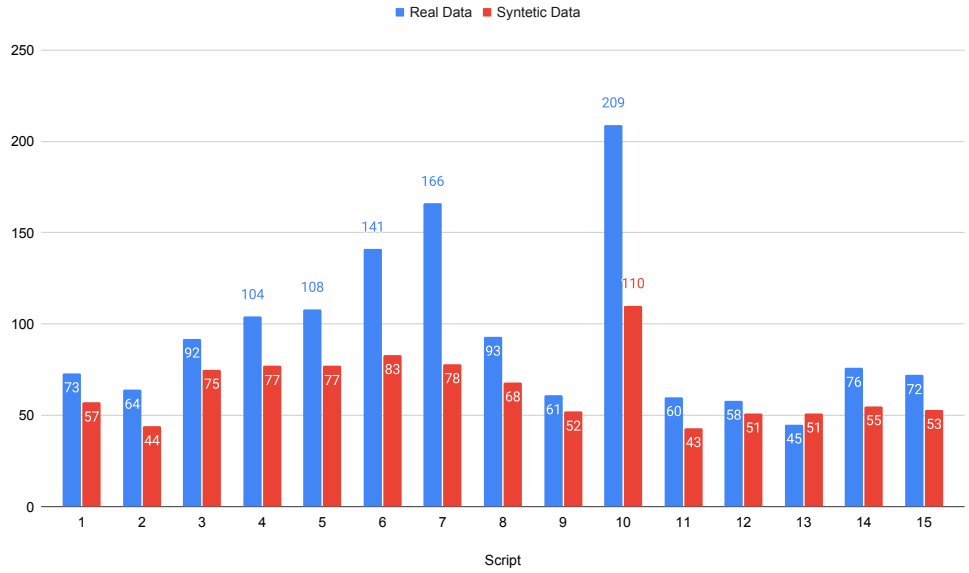


Figure 7.7 – Number of sensor activations per scenarios

The Figure 7.8 show a comparison of the number of sensor triggered during the scenario. We can observe that each scenario use a same number and equivalent sensor, except for the scenario 5 and 13. In scenario 5 one more sensor was triggered in the real data while in scenario 13 two more sensors were triggered in the synthetic data.

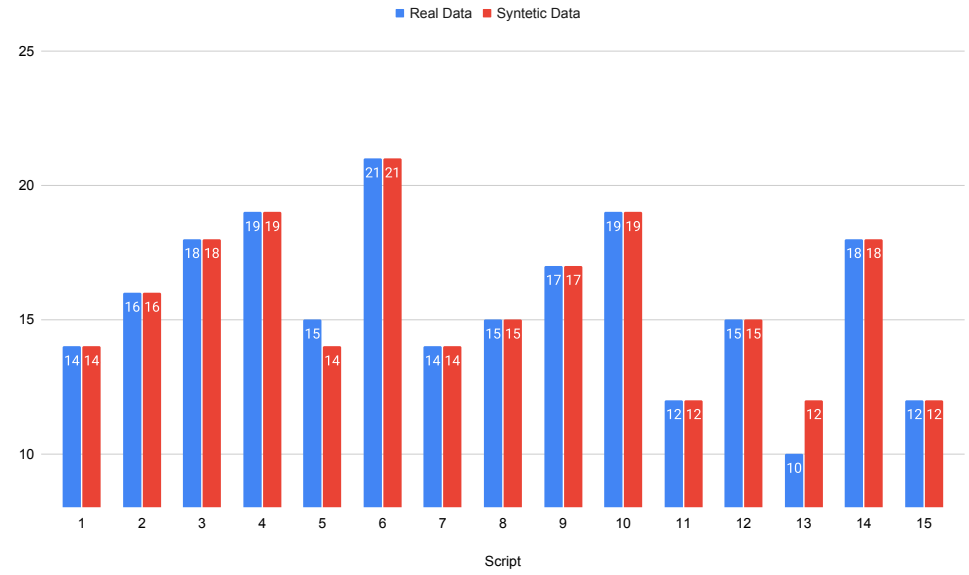


Figure 7.8 – Number of unique sensor per scenarios

Comparison with Cross-Correlation Method

In the context of a digital twin, it is important to evaluate whether the data generated by the simulator are equivalent to the real data. Moreover the similarity between synthetic and real data must be respected so that an algorithm can be trained under the same conditions as if it had been trained on real data. In order to compare more precisely the synthetic and real data, we used the Cross-Correlation Method [12].

Cross-Correlation is used in signal processing, which measures the similarity of two series as a function of the displacement of one relative to the other. This is also known as a sliding dot product or sliding inner-product. The cross-correlation is similar in nature to the convolution of two functions. In an autocorrelation, which is the Cross-Correlation of a signal with itself, there will always be a peak at a lag of zero, and its size will be the signal energy. The Cross-Correlation for discrete function [10] is expressed as in Equation 7.1. Where f and g are two functions, and n is the displacement or lag between the two functions.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} \overline{f(m)} g(m+n) \tag{7.1}$$

Table 7.2 contains the cross correlation score for a lag of 0 between each synthetic and real scenarios. The results show that synthetic dataset seem close, which means that the simulator can provide sensor logs close to reality (between 70.00% and 97.00%). We can observe the worst similarity for the scenario 14. In the next section, we will use these data to train an ADLs recognition algorithm and conclude if Digital Twin data can be used.

Table 7.2 – Cross correlation similarity

Subject	S9						S7					S3			
Scenario Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Similarity (%)	75,73%	97,40%	75,42%	75,30%	69,61%	74,42%	81,24%	93,18%	85,22%	84,73%	88,28%	79,93%	77,53%	54,73%	82,78%
Average Similarity (%)	77,98%						86,53%					73,74%			

7.6 Human Activity Recognition in Smart Homes from Digital Twin Data: Algorithm Training and Evaluation

7.6.1 The Algorithm

In order to, recognize the ADLs in the smart apartment we choose to use the Liciotti's [3] bidirectional LSTM model. This choice is motivated by the size of the dataset. Indeed, we reproduced very few data. Thus, as the data quantity is low, a simpler model is preferred. The usage of the ELMo proposed model requires much more data to extract relevant features and generalise well.

7.6.2 The Evaluations

To validate the generated data, we trained the algorithm following four methods and experiments explained in the following subsections. To perform the training we proceeded as in the previous chapters, i.e. we segmented the data in the form of activity sequence. We also grouped under the same label "Cook" all activities related to cooking, i.e. "Cook breakfast", "Cook dinner" and "Cook lunch". We did the same for activities related to the more generic activity "Eat", i.e. "Eat breakfast", "Eat dinner", "Eat lunch". This grouping is motivated by the fact that the real experiments were carried out during the day and do not respect the wall clock time of appearance of each scenario. In other words, the "evening" scenario was not realized in the evening, and the same is true for the others. This prevents us from using temporal information to differentiate these activities, which are very strongly linked to the time of day. For the training of the algorithms, we used each time 20% of the training set as validation data to stop the training in case of overfitting (see Chapter 3).

Leave One Subject Out Cross Validations

The first two approaches apply the Leave One Out cross validation method [11] on the one hand on the synthetic data only and on the other hand on the real data only. This type of methods is usually used for datasets containing small amounts of data or when the data are linked to whole entities. As we have three subjects, the principle is to use one subject for the test and the two others for the training. Then a rotation on which subject becomes the test set is performed which allow us to validate the algorithm on each subject data. Finally, we make a comparison between results of the synthetic and real Leave One Out cross validation.

Table 7.3 – Results of Leave One Out Cross Validation on real and synthetic data for each subject

	Real			Synthetic		
Subject	S9	S7	S3	S9	S7	S3
Accuracy	71,43	51,36	71,43	73,47	72,98	67,86
Precision	80,34	57,25	62,67	87,22	73,64	77,00
Recall	80,00	55,84	65,84	81,58	68,34	75,00
F1-score	75,64	52,5	60,74	79,53	66,02	75,8
Balanced Accuracy	80,00	55,83	65,83	81,57	68,33	75,00
Weighted Precision	77,08	54,19	68,34	85,66	73,41	70,72
Weighted Recall	71,43	51,36	71,43	73,47	72,98	67,86
Weighted F1-score	70,12	49,19	65,88	71,65	68,20	69,00

The Table 7.3 provides the result of the two Leave One Out cross validations. The results demonstrate that the algorithm can be trained with real and synthetic data and obtain almost similar results. We can observe a high similarity between the results of the subject "S9". However,

we can see on the contrary more difference between results of synthetic and real data for the two other subject. The performance are better for the subject “S7” and “S3” with synthetic data according to the F1-score and the Balanced Accuracy.

7.6.3 One to One

In this experiment, we train the model with the synthetic data of a subject and test the trained algorithm with the real data of the same subject. The result should allow us to have a first evaluation on the usability of the synthetic data. The results are presented in Table 7.4.

At first sight, the synthetic data generated for each subject allows to train and recognize activity sequences for each real dataset (one subject step). Subjects “S9” and “S7” obtain good performances in terms of Accuracy, Balanced Accuracy, and F1-score. We notice that subject “S7” presents the best performances. For these two subjects the synthetic data seems to be realistic enough to allow a recognition of more than 70% in terms of accuracy for both subjects.

On the contrary, the “S3” subject presents the worst performances. It seems that the synthetic data generated for this subject, are not able to allow a recognition of the activities of this subject.

We notice for the subject S9 that the activity “Bathe” is not recognized, while for the subject “S7” it is recognized at 100%. The subject “S3” presents 4 classes of activity out of 10 not recognized. From these results we can deduce that it is possible to use synthetic data to train an algorithm and recognize activities from real data. However, it seems that using only activity data from a single subject is not always sufficient. It is therefore necessary to bring more data and variability in the training dataset.

Table 7.4 – Results of one to one experiment

	S9				S7				S3			
	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
Bathe	0,00%	0,00%	0,00%	2	100,00%	100,00%	100,00%	2	50,00%	100,00%	66,67%	4
Cook	90,00%	90,00%	90,00%	10	54,55%	100,00%	70,59%	6	75,00%	75,00%	75,00%	4
Dress	75,00%	100,00%	85,71%	3	100,00%	100,00%	100,00%	3	100,00%	50,00%	66,67%	2
Eat	50,00%	50,00%	50,00%	6	100,00%	75,00%	85,71%	4	0,00%	0,00%	0,00%	3
Enter Home	50,00%	75,00%	60,00%	4	100,00%	33,33%	50,00%	3	60,00%	100,00%	75,00%	3
Go To Toilets	66,67%	80,00%	72,73%	5	75,00%	100,00%	85,71%	3	0,00%	0,00%	0,00%	2
Leave Home	40,00%	50,00%	44,44%	4	60,00%	100,00%	75,00%	3	0,00%	0,00%	0,00%	2
Sleep	50,00%	33,33%	40,00%	3	100,00%	75,00%	85,71%	4	100,00%	50,00%	66,67%	2
Wash Dishes	100,00%	80,00%	88,89%	5	100,00%	25,00%	40,00%	4	44,44%	100,00%	61,54%	4
Watch TV	100,00%	85,71%	92,31%	7	100,00%	80,00%	88,89%	5	0,00%	0,00%	0,00%	2
Accuracy	71,43%				78,38%				57,14%			
Balanced Accuracy	64,40%				78,83%				47,50%			
Macro Avg	62,17%	64,40%	62,41%	49	88,95%	78,83%	78,16%	37	42,94%	47,50%	41,15%	28
Weighted Avg	70,78%	71,43%	70,39%	49	87,36%	78,38%	76,91%	37	44,92%	57,14%	46,59%	28

7.6.4 Many to One

In this experiment, the algorithm is trained on all synthetic data. Once trained, it is tested independently with the real data of each subject. The purpose of this experiment is to evaluate

whether increasing the amount of data provided by the synthetic data of the different subjects improves the capabilities of the algorithm for recognizing the ADLs. Table 7.5 shows the results of this experiment.

Table 7.5 – Results of the many to one experiment

	S9				S7				S3			
	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
Bathe	100,00%	100,00%	100,00%	2	100,00%	100,00%	100,00%	2	100,00%	75,00%	85,71%	4
Cook	100,00%	80,00%	88,89%	10	50,00%	100,00%	66,67%	6	100,00%	25,00%	40,00%	4
Dress	100,00%	100,00%	100,00%	3	100,00%	66,67%	80,00%	3	100,00%	100,00%	100,00%	2
Eat	50,00%	66,67%	57,14%	6	100,00%	50,00%	66,67%	4	100,00%	66,67%	80,00%	3
Enter Home	66,67%	50,00%	57,14%	4	100,00%	100,00%	100,00%	3	75,00%	100,00%	85,71%	3
Go To Toilets	100,00%	80,00%	88,89%	5	100,00%	100,00%	100,00%	3	66,67%	100,00%	80,00%	2
Leave Home	100,00%	75,00%	85,71%	4	100,00%	100,00%	100,00%	3	100,00%	50,00%	66,67%	2
Sleep	37,50%	100,00%	54,55%	3	66,67%	100,00%	80,00%	4	100,00%	100,00%	100,00%	2
Wash Dishes	100,00%	80,00%	88,89%	5	0,00%	0,00%	0,00%	4	57,14%	100,00%	72,73%	4
Watch TV	100,00%	85,71%	92,31%	7	100,00%	80,00%	88,89%	5	66,67%	100,00%	80,00%	2
Accuracy	79,59%				78,38%				78,57%			
Balanced Accuracy	81,74%				79,67%				81,67%			
Macro Avg	85,42%	81,74%	81,35%	49	81,67%	79,67%	78,22%	37	86,55%	81,67%	79,08%	28
Weighted Avg	87,33%	79,59%	81,67%	49	77,48%	78,38%	74,89%	37	86,44%	78,57%	76,58%	28

We can observe that the performances, compared to the One to One method, are improved. Except for the subject “S7” which keeps very similar performances to the previous experiment. However, for subjects “S9” and “S3” we observe an improvement, especially for subject “S3”, who until now showed the worst results. These results show us that it seems conceivable and possible to generate fully synthetic data, to use them to train an algorithm and to recognize activity sequences from real data.

7.7 Summary

The datasets are not sufficient to cover the distribution of possibilities in terms of activities, lifestyles, or house configurations. They are limited in the number of activities, occupants they consider, as well as the types of sensors and the category of residents. Therefore, algorithms trained on data from one particular environment will be difficult or impossible to reuse in others, as they are tailored to a particular home configuration, inhabitants, and living habits.

Collecting new datasets to increase the number of examples is a very expensive task in terms of equipment and difficult to achieve. It is necessary to ask residents to live in the environment and label their activities over several weeks or months. This involves a significant investment on the part of volunteers and presents a high risk of mislabeling.

The use of transfer learning techniques seems to be a solution. Nevertheless, it is necessary to have labeled data from the target environment to adjust the algorithms to fit perfectly. From the perspective of a real-world business application, it is not feasible to ask end users or customers to label their activities themselves for several days, weeks, or months.

To address the challenges of data scarcity and, in particular, the gap between training data and use case data, in this chapter, we proposed data generation through the digital twin concept. We improved and extended the Virtual Home simulation environment to allow it to simulate an intelligent environment equipped with home automation sensors. We reproduced a digital version of a real intelligent apartment in this simulation environment. Then, with the help of volunteers, we produced data in the real intelligent apartment through activity scenarios. We finally reproduced the volunteers' activities, via an avatar, in the Digital Twin of the apartment. We were able to see through analysis and use of similarity metrics between time series, that the data generated by the simulator and that generated by the volunteers are relatively similar.

Thanks to this, we were able to generate synthetic sensor logs, which allowed us to train a classification algorithm for ADLs. This algorithm could be validated on the real data created by the volunteers. We were able to demonstrate that the algorithm was able to recognize activities on real data with good performances (around 80% on average according to the F1-score metric). However, the experiments were performed on a small amount of data. It would be necessary to validate the results through a larger amount of data. But also from data generated with residents living perpetually in the environment in question. We can nevertheless conclude that the use of a Digital Twin to generate data similar to the target environment, in order to train an activity recognition algorithm seems promising.

BIBLIOGRAPHY

[1] Experiment'Haal, le Living lab santé autonomie (LLSA). URL `http://www.imt-atlantique.fr/fr/recherche-et-innovation/plateformes-de-recherche/experiment-haal`.

[2] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: definitions, characteristics, applications, and design implications. *IEEE access*, 7:167653–167671, 2019.

[3] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. CASAS: A Smart Home in a Box. *Computer*, 46(7):62–69, July 2013. ISSN 1558-0814. doi: 10.1109/MC.2012.328. Conference Name: Computer.

[4] Anthony Fleury, Michel Vacher, François Portet, Pedro Chahuara, and Norbert Noury. A french corpus of audio and multimodal interactions in a health smart home. *Journal on Multimodal User Interfaces*, 7(1):93–109, 2013.

[5] Sidney Katz. Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31(12):721–727, 1983.

[6] Siavash H Khajavi, Naser Hossein Motlagh, Alireza Jaribion, Liss C Werner, and Jan Holmström. Digital twin: vision, benefits, boundaries, and creation for buildings. *IEEE access*, 7:147406–147419, 2019.

[7] Paula Lago, Fréderic Lang, Claudia Roncancio, Claudia Jiménez-Guarín, Radu Mateescu, and Nicolas Bonnefond. The contextact@ a4h real-life dataset of daily-living activities. In *International and interdisciplinary conference on modeling and using context*, pages 175–188. Springer, 2017.

[8] Afiqah Ngah Nasaruddin, Teruaki Ito, and Tee Boon Tuan. Digital twin approach to building information management. In *The Proceedings of Manufacturing Systems Division Conference 2018*, page 304. The Japan Society of Mechanical Engineers, 2018.

[9] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[10] Lawrence R Rabiner and Bernard Gold. Theory and application of digital signal processing. *Englewood Cliffs: Prentice-Hall*, 1975.

[11] Claude Sammut and Geoffrey I. Webb, editors. *Leave-One-Out Cross-Validation*, pages 600–601. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_469. URL https://doi.org/10.1007/978-0-387-30164-8_469.

[12] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *Circuits, systems and signal processing*, 28(6):819–843, 2009.

CONCLUSIONS AND PERSPECTIVES

8.1 Conclusion and Contributions

Today, due to the reduction of manufacturing costs, the progress of sensors and communication systems, our environment and our daily life is increasingly equipped with sensors and intelligent interconnected objects. This technology allows our environment to become more intelligent and capable of providing multiple new services. The smart home is at the center of attention for the many innovative applications and services it can offer in terms of security, energy saving, comfort improvement and health support.

The last few years have seen the emergence of a multitude of techniques and approaches through artificial intelligence to equip homes with the cornerstone to all these services; the ability to understand the lifestyle and activities of its residents through home automation sensors. Despite advances in deep learning and the increase in the amount of data, these methods do not allow for sufficient generalization and full treatment of the complexity and variability of human activity. This observation allowed us to question these limitations and to try to answer them in order to propose new ways of improvement and possible research directions. We have therefore tried to answer the following questions:

- ***How to efficiently encode, represent, and extract information, features, and patterns from non-intrusive home automation sensors of different natures to perform recognition of the ADLs?***
- ***How can we help recognition algorithms to better understand the sequences of sensor activations and sensor relatedness to separate the sequences and better understand captured sensor traces?***
- ***Can we transfer the knowledge gained from one house to another house?***
- ***How could we apply activity recognition methods to an actual real house?***

8.1.1 How to efficiently encode, represent, and extract information, features, and patterns from non-intrusive home automation sensors of different natures to perform recognition of the ADLs?

The actual human activity recognition approaches have difficulties because home automation sensors are voluntarily simple and not very rich in order to information to respect the privacy of the resident(s) and to save energy. The values they provide can also be of very different nature (binary, scalar, states, etc.). Moreover, the time series generated by these sensors are event-driven because they are triggered by the actions of the residents or the environment. This makes these time series not sampled in a regular way, requiring a particular treatment.

For this purpose, we proposed to treat these time series as a single sequence of events and to be inspired by methods of representation and sequence processing such as those used in the field of Natural Language Processing. We used embedding methods that allowed us to obtain rich representations of these events and their relations. We have tried to couple this representation with a time series classification method. We have compared a LSTM approach, designed to analyse sequential data and which have demonstrated exemplary performance, against the FCN algorithm, which has also shown good performance for classical time series classification.

Despite the ability of FCN to extract features and patterns, our results (cf. Chapter 4) have allowed us to demonstrate: 1) that the sequential representation and the use of an embedding allows to improve the classification performances of the models 2) that the problem of activity recognition is not only a pattern recognition problem and requires in addition to interpret temporal and potentially contextual dependencies. Indeed, the patterns in the activity sequences are not exactly similar between sequences belonging to the same label due to sampling in the form of events. This form of sampling imposed by the rhythm of life of the habitats distorts the patterns because the events do not appear exactly with the same temporal spacing or regularity.

8.1.2 How can we help recognition algorithms to better understand the sequences of sensor activations and sensor relatedness to separate the sequences and better understand captured sensor traces?

Human beings do not always live alone. Because of actions done by several residents in parallel, the activation trace of a sensor corresponding to an activity can be mixed with another trace. By interpreting that certain orders of actions are possible or not, it is easier to distinguish what each resident is doing. For example, let us consider two residents, each one performing a different activity. The first one cooks while the second has a shower. In this case, the sensors of the house could potentially capture the actions of each resident in the following order: presence in the kitchen, presence in the bathroom, the fridge door open, the bathroom light on, the kitchen faucet, the hotplate on, the shower faucet on , the kitchen faucet off, etc. It is therefore useful

to be able to interpret the logic of the possible sequences of actions. Moreover, in the case of a resident living alone, this knowledge should allow to better contextualize and interpret similar activities in terms of sensor sequence as involving the same sensors.

In order to better interpret the sequences of sensor activations, their relations and the logic of their sequencing, we once again relied on the language paradigm and NLP techniques, in particular on language modeling. We trained and used two advanced forms of embeddings based on Word2Vec and ELMo. By confronting these methods, we were able to observe and demonstrate that the use of a contextualized embedding such as ELMo increases the classification performance of a model based on a bidirectional LSTM. We have noted that taking into account the context and modeling the probability of occurrence of a sensor activation, has particularly improved the performance of the classifier on a dataset containing the activities of several people.

Another important aspect of human activity to consider is time. Human life is very often correlated with the wall-clock-time or periods of the day.

To account for the time of the day, we added an encoding and an interpretation of time (cf. Chapter 6). We could observe that the addition of the time in the form of sequences allowed to increase the performances of the model coupling the contextual embedding ELMo and a classifier based on a bidirectional LSTM. We also noticed that this approach drastically increased the performance of the Liciotti et al. [3] model that was used as a baseline, especially on the Cairo multi-resident dataset.

In addition to often having a temporal aspect, ADLs are often correlated with each other. Some activities are expected to appear before or after others. For instance, the activity of cooking calls for the activity of eating. This is why we tested the extension of the context of an activity sequence by adding the previous activities. However, the approach using the previous activities did not lead to any improvement. We even noticed a performance degradation except on some particular activities depending on the datasets. We suppose that the degradation comes from the size of the input sequences, which have become potentially too long in number of events for the LSTM based model.

8.1.3 Can we transfer the knowledge gained from one house to another house?

Houses have very different typologies, resulting in even more sensor configurations. The advantages of deep learning-based methods are their adaptability and generalizability. The increase in the number of datasets and thus access to rich data with multiple activity labels, as well as the increase in computational power, make it possible to design and generate more complex models today. But while a deep learning model can interpolate a model on a specific dataset, its performance drops when tested on another dataset. Thus, due to the variability of houses and configurations in each dataset, interpolation shows it limit, and we would need a mechanism

such as transfer learning to enable better generalisation of models.

The NLP domain uses in particular pre-trained embedding models for generalizing on a set of data and transferring its knowledge to a new context. We were inspired by this method to transfer the knowledge acquired on the Aruba dataset to reuse it on a second dataset, Cairo, through the ELMo embedding. We trained a new classifier on the Cairo dataset using the ELMo embedding of Aruba. We observed that the knowledge and representation of the embedding from Aruba allowed us to obtain very similar results to training a classifier with an embedding trained on the destination dataset. The use of pre-trained embedding allows to transfer knowledge and representation from a house A to a house B in the same way as in the NLP domain for text. It would therefore be possible to have smaller destination datasets but to take advantage of the generalization from larger datasets via pre-trained embedding.

8.1.4 How could we apply activity recognition methods to a actual real house?

The use and application of activity recognition approaches in a real-world, industrial and commercial context remains a major challenge due to the lack of labeled data from the destination environment. Although activity recognition models can be trained on any dataset, it is still impossible to obtain a large amount of labeled data from a client. It is not realistic to collect several months of data and ask a client to label it.

One idea, increasingly common in manufacturing or industry, and now in the medical field, is the creation and use of digital twins. The intuition is to be able to apply, study, observe on a digital double certain changes or effects. This intuition extends to real time changes that can come from the physical or virtual world. Inspired by this idea of digital replica, we tried to answer the problem of transferability of activity recognition methods to the real world.

We created a digital copy (Digital Twin like), containing virtual sensors, of a real intelligent apartment that we used to generate a synthetic data set. For this we used an avatar, simulating a resident, to perform activities of daily living in this simulation environment. The data from the virtual sensor activations recorded during the execution of the avatar's activities were used to train an activity recognition algorithm. We showed that the generated data was similar to the real data of the apartment. At the same time, we showed that the algorithm trained on synthetic data obtains high classification performances ($\tilde{8}0\%$) on the real apartment data. This work has shown that it is possible to train an algorithm from synthetic data and label it without acquiring and labeling real data.

8.2 Limitations and Perspectives

In this thesis, we have sought to improve the performance of recognition algorithms through the language paradigm and techniques from the NLP domain. We have shown that contextualization and the consideration of time allow us to push back certain limits. We have seen that it was possible to transfer knowledge from one house to another, but also to transfer knowledge from a virtual house to a real one. However, limitations remain in the proposed approach and further work could be considered to address them. Furthermore, based on this preliminary work, new perspectives can be explored.

8.2.1 New Datasets Comparison

A limitation of this work is the evaluation of the proposed method on only three CASAS datasets. An interesting perspective would be to apply the method in a first step on a larger panel of datasets provided by the CASAS benchmark [3]. We could think of the series of datasets "HH" containing thirty similar apartments. Then in a second step, on other datasets available in the literature such as, [5], [13], [1], etc. It could be observed the generability of the method proposed in this thesis, as well as potential new limitations.

8.2.2 No Pre-Segmented Activity Recognition and Activity Sequences Auto-Segmentation

This work is based on the classification of EW, i.e. of pre-segmented activity sequences. The model thus has as input all the sensor activations corresponding to the activity to decide. It knows the beginning, the end and the length (in terms of number of events to estimate the duration). However, in order to create more dynamic applications and services, it is necessary to recognize the ADLs during and as the residents perform them.

A first perspective of work would be to transpose and evaluate the proposed method in a “online” activity recognition context. A first approach would be to train the method as realized here, on EWs. Then to use TWs for the final recognition application. A second possibility would be to train the embedding on EW but to train the classifier on TW. The classifier could then exploit the knowledge learned on the dataset of activity sequences to try to recognize each TW.

Concerning the ELMo embedding training, it is necessary to have labeled dataset. This embedding is trained with independent sentences and thus, need segmentation in EW. This cutting into EW avoid the understanding of the relations between activity sequences. However, it could be considered not to use EW for embedding training and to exploit an approach similar to GPT [14]. GPT follows the basic principle of language modeling by predicting the next word from the previous words. But unlike ELMo or BERT, the preceding words are potentially words belonging to the previous sentence. This allows the interpretation of a more global context, but

potentially allows finding breaks between sentences thanks to context changes. It could be the same for activity sequences.

8.2.3 Dealing with Unknown Sensors Values

In this work, we used an encoding of sensor events in the form of a categorical variable sequence (concatenation of the sensor ID and its value). This encoding method is limited by a certain size of "vocabulary" or, in other words, the possible sensor activation from the training data. It is currently impossible to obtain a representation of the values of sensors that have never been observed in training. For example, a temperature sensor that never exceeded 30 degrees in the training dataset but has a higher temperature in real-world model use will not be represented. Even though we have prevented this case by using, as it is done in the NLP domain, a special word ,"<UNK>" to replace the unknown values, too many unknown values in the data would lead to a degradation of the classification performance. The simplest solution is to obtain the largest possible volume of data to cover the maximum possible values, but getting a large amount of data with a variety of examples is not always easy.

The NLP field has already considered and proposed solutions that could be exploited. We could exploit a character-based representation [16], using the version of the ELMo model based on word embedding from a CNN. But also, methods such as Byte Pair Encoding (BPE) [10] or WordPiece [15] could be considered to split words representing a sensor activation, into character compositions or subwords. The use of WordPiece seems a promising option as we could consider a sensor activation in two words, "T001" and "30", which will bring several advantages.

Firstly, using the example above, only the value of 31 degrees would be replaced by the special word "<UNK>" and not the entire sensor activation. This leaves the model with the ability to interpret which known sensor activates, regardless of its value.

Secondly, as the training of an embedding based on language modeling seeks to predict the next token (word), in the case of using subwords, the tokens would represent subwords. The model would then be trained to predict the subwords which would add the possibility for the model to learn that, a value of a temperature sensor is necessarily scalar and can only evolve between -40 and +50 degrees for example. This would add an additional level of knowledge, potentially useful for the detection of anomalies or impossible values.

8.2.4 Transfer Learning, Universal Smart Home Language and Translation

We have shown that the use of a pre-trained embedding allows to transfer knowledge and representation of sensors from one house to another. However, in our experiment, we have realized this transfer between two datasets belonging to the CASAS benchmark. The two datasets share a very similar vocabulary as the naming of the sensors and their values follows the same structure. Thus, a word representing the activation of a sensor present in dataset "A" has a very high

chance of appearing in the vocabulary of dataset "B".

If this same experiment were performed between two datasets using different naming conventions, the transfer learning would need another step, indeed each of the two houses has different "vocabularies". To echo the NLP domain, it would be like trying to transfer the knowledge learned on an English corpus to a French corpus.

Moreover, the proposed encoding lacks of universality. Even in the case where the "vocabulary" is similar, the name or ID of a motion sensor located in the kitchen for dataset "A" can designate a completely different sensor in dataset "B", a motion sensor in the bathroom for example. Even if we have observed during our experimentation that this limitation does not seem to have a strong impact, being able to keep the "sense" and the context of activation of a sensor learned in one dataset, could allow to guarantee better performance in the transfer of knowledge to another dataset. At least two solutions could be considered to overcome these limitations.

First, by creating a universal vocabulary so that a sensor, such as a motion sensor in the kitchen, always has the same name, or naming strategy, regardless of the dataset.

Secondly, it would be possible to draw inspiration from translation models and methods based on Sequence to Sequence (Seq2Seq) models [11], [2]. Intuitively, considering that each house has its own "language", it might be possible to translate the activity sequences of house A into the "language" of house B.

8.2.5 Transformers-Based Embedding

In this work, we used the embedding methods Word2Vec and ELMo to demonstrate the feasibility and interest of transferring these methods from the NLP field to the domain of human activity recognition in smart homes. However, new methods based on a structure called Transformers [14] have emerged in recent years in the NLP domain. Thanks to their ability to capture distant dependencies, in theory infinitely, and to focus attention on important elements in sequences, methods based on Transformers have become state of the art. They have for the same reason become state of the art and relevant solutions in other domains such as Music Generation [4] and Music Classification [12], which also deal with sequence problems.

Transformers could be a new model to exploit for the human activity recognition in smart homes, in particular by using embedding methods today state of the art, such as GPT [14] and BERT [4]. GPT would allow a better interpretation of the long and very long term dependencies of the sensors, while BERT would allow to reinforce the understanding of the local context (internal to the sequence of activity) but also inter-sequences of activity relations context.

Indeed, BERT is trained in parallel according to two tasks: the Masked Language Model (MLM) task, for the contextual understanding of words; and the Next Sentence Prediction (NSP) task, for understanding relations between sentences. The NSP task consists in training the model to predict whether two sentences follow each other. In the case of ADLs recognition,

it would be to predict whether two activities follow each other. This would make it possible to exploit the logic of the succession of activities to better recognize them.

8.2.6 Digital Twin and Synthetic Data Generation

In order to realize the digital twin of a connected apartment, we augmented the Virtual Home [6] simulation environment to create VirtualSmartHome. Although this simulator allowed us to set up a proof of concept, there remains a main limitation. Indeed, Virtual Home is built on top of the Unity3D set engine and does not offer the possibility to accelerate the simulation environment. Unity3D is used as a real time rendering engine. It is therefore not suitable for such an operation. Moreover, the version of VirtualSmartHome remains limited in terms of actions, interactions and behavior modeling.

A possible evolution path for the creation of a new version of VirtualSmartHome could be the exploitation of video games environments. Indeed, video games are a multi-billion dollar industry, where developers are striving to build very realistic worlds. Always associated with entertainment, the impressive amount of detail and diversity of objects, environments, appearances, human body movements and character traits, as well as the power to control the virtual world make it an excellent source of training data.

This idea has already been considered as a data source for semantic segmentation in autonomous vehicle applications [7], [8] but also ADLs video generation for training robots [9]. This same idea could be considered for the generation of sensor data, through life simulation sets, and thus take advantage of the richness of these environments. Moreover, today some video sets have APIs to allow the creation of personal extensions.

"Progress in knowledge is essentially the transformation of previous knowledge."

– Karl Popper, *On the Sources of Knowledge and of Ignorance*

BIBLIOGRAPHY

[1] Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. Aras human activity datasets in multiple homes with multiple residents. In *2013 7th Inter. Conf. on Pervasive Computing Technologies for Healthcare and Workshops*, pages 232–235. IEEE, 2013.

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] D.J. Cook and M. Schmitter-Edgecombe. Datasets. URL `https://tinyurl.com/bdd5hkph`.

[4] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[5] Fco Ordóñez, Paula De Toledo, and Araceli Sanchis. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13 (5):5460–5477, 2013.

[6] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[7] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[8] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017.

[9] Alina Roitberg, David Schneider, Aulia Djamal, Constantin Seibold, Simon Reiß, and Rainer Stiefelhagen. Let's play for action: Recognizing activities of daily living by learning from life simulation video games. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8563–8569. IEEE, 2021.

[10] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[12] TJ Tsai and Kevin Ji. Composer style classification of piano sheet music images using language model pretraining. *arXiv preprint arXiv:2007.14587*, 2020.

[13] Tim LM van Kasteren, Gwenn Englebienne, and Ben JA Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity recognition in pervasive intelligent environments*, pages 165–186. Springer, 2011.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[16] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

LIST OF PUBLICATIONS

[1] D. Bouchabou, S. M. Nguyen, C. Lohr, B. LeDuc, and I. Kanellos, "Fully convolutional network bootstrapped by word encoding and embedding for activity recognition in smart homes," in *International Workshop on Deep Learning for Human Activity Recognition*. Springer, 2021, pp. 111–125.

[2] ——, "A survey of human activity recognition in smart homes based on iot sensors algorithms: Taxonomies, challenges, and opportunities with deep learning," *Sensors*, vol. 21, no. 18, p. 6037, 2021.

[3] ——, "Using language model to bootstrap human activity recognition ambient sensors based in smart homes," *Electronics*, vol. 10, no. 20, p. 2498, 2021.

[4] ——, "Human Activity Recognition (HAR) in Smart Homes," p. https://encyclopedia.pub/17108, Nov 2021, MDPI - Scholarly Community Encyclopedia. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03436948

[5] ——, "Reconnaissance d'activités de la vie quotidienne au moyen de capteurs domotiques : lorsque syntaxe, sémantique et contexte se rencontrent." *Revue Ouverte d'Intelligence Artificielle*, 2021.

[6] ——, "What if time, or past context, improved human activity recognition in smart homes?" in *IEEE ICCE 2022 - 9th International Conference on Communications and Electronics*, 2022.

[7] J. Grosset, D. Bouchabou, S. M. Nguyen, and C. Lohr, "A smart home digital twin to support the recognition of activities of daily living."

Appendix A

Experiments and Models Hyperparameters

Table A.1 – General hyperparameters

K cross validation	3
Max sequence lenght	2000
Max epochs number	400
Batch size	64
Patience	20

Table A.2 – Embeddings hyperparameters

	Embedding	Word2Vec	ELMo
Embedding size	64	64	64
Context windows size	None	60	60
Max epochs number	400	100	400
Batch size	None	None	512

Table A.3 – Classifiers hyperparameters

	LSTM	Bi LSTM
Nb Units	64	64
NB layers	1	1

Appendix B

Sensor2Vec Embedding Vizualisation

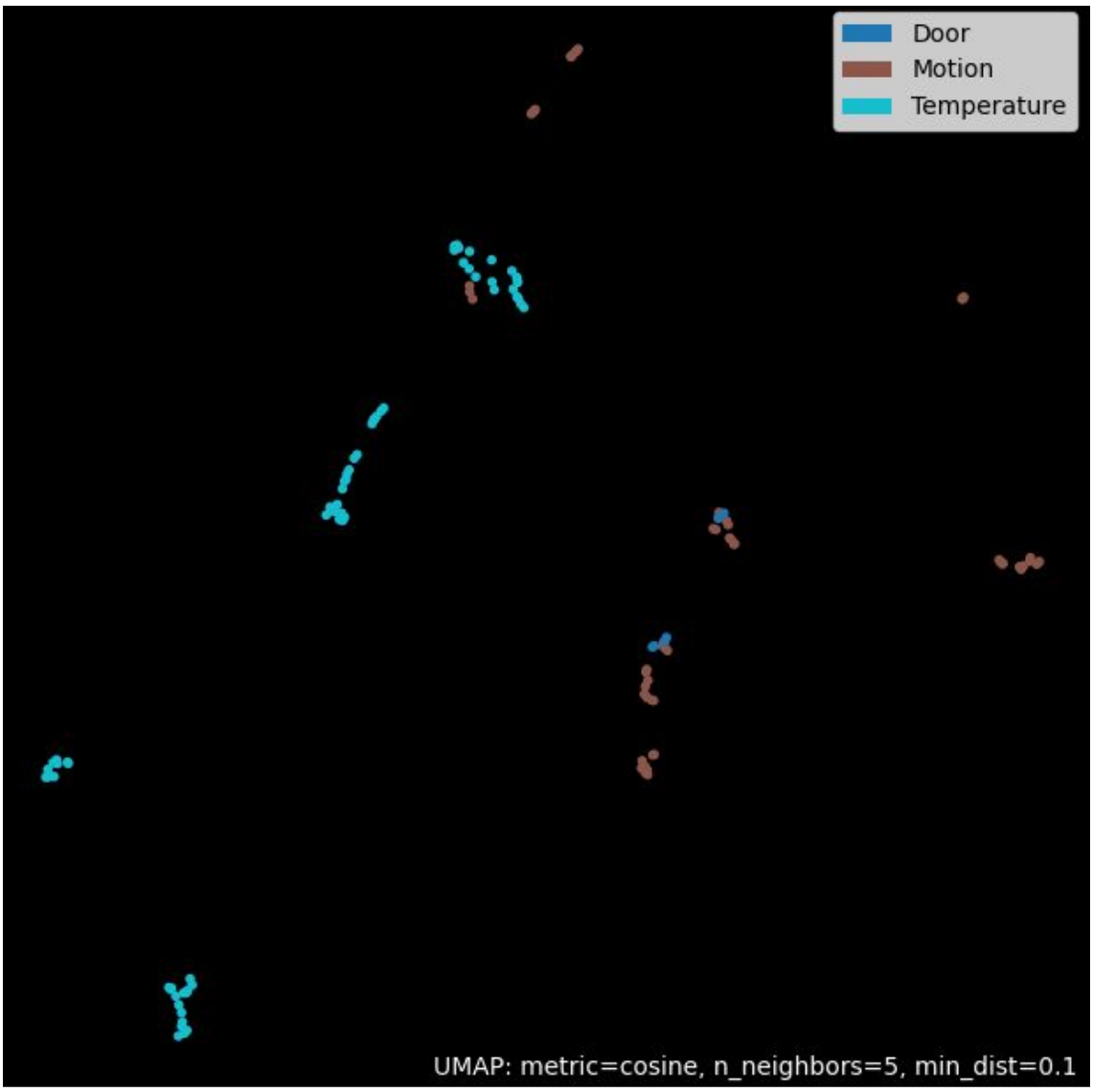


Figure B.1 – Milan's cluster colorized by sensors' nature

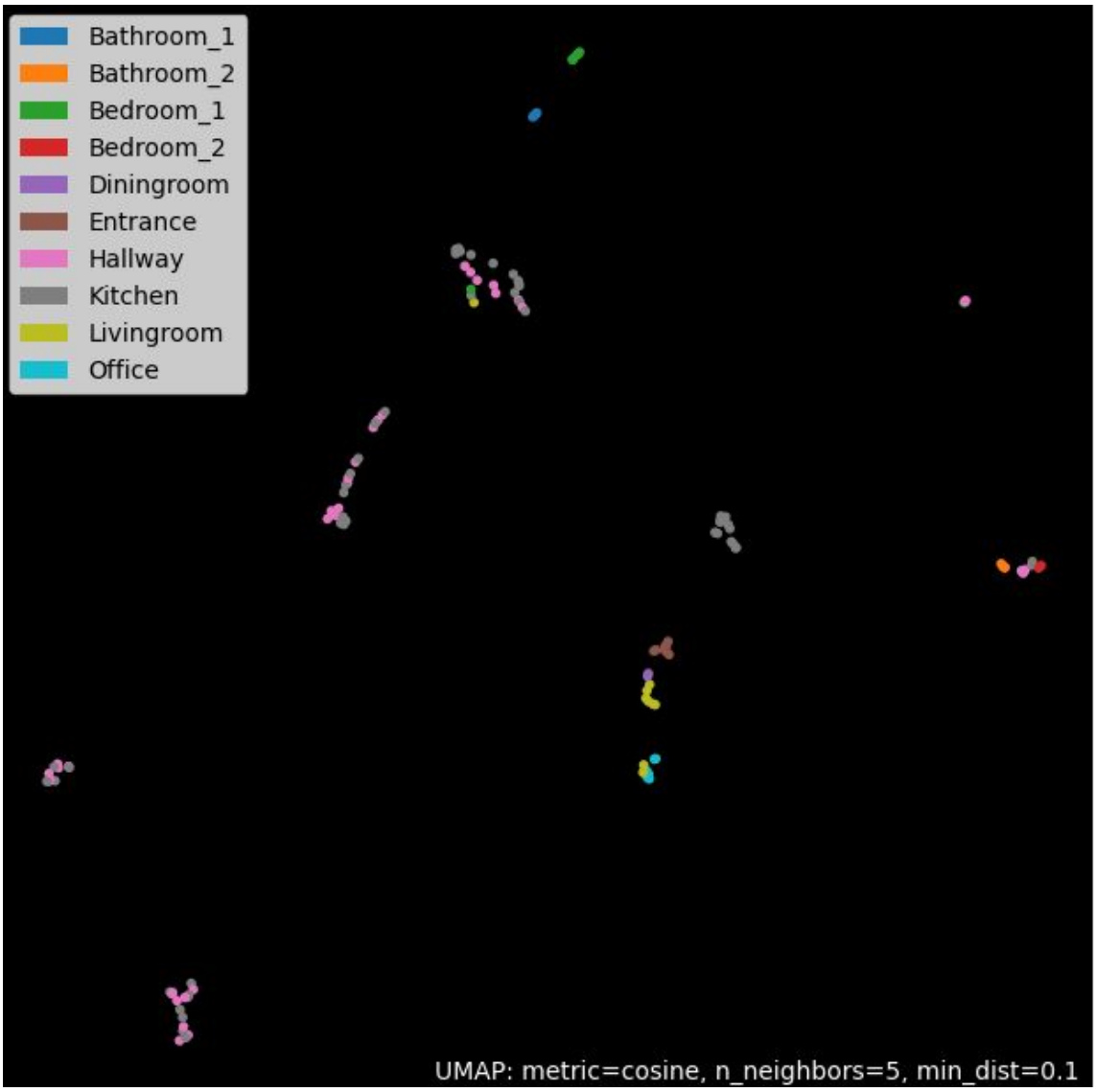


Figure B.2 – Milan's cluster colorized by sensors' localisation

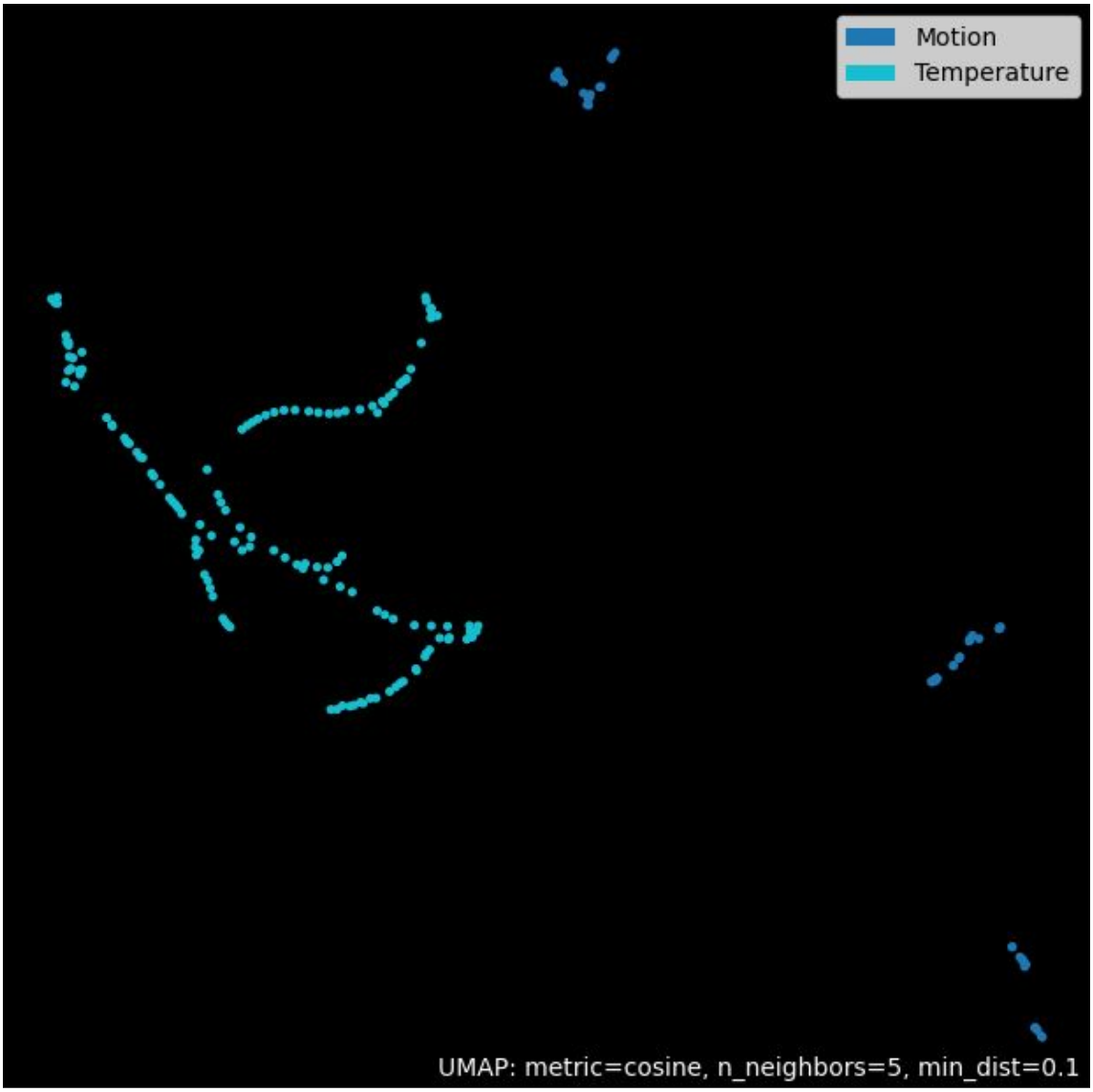


Figure B.3 – Cairo's cluster colorized by sensors' nature

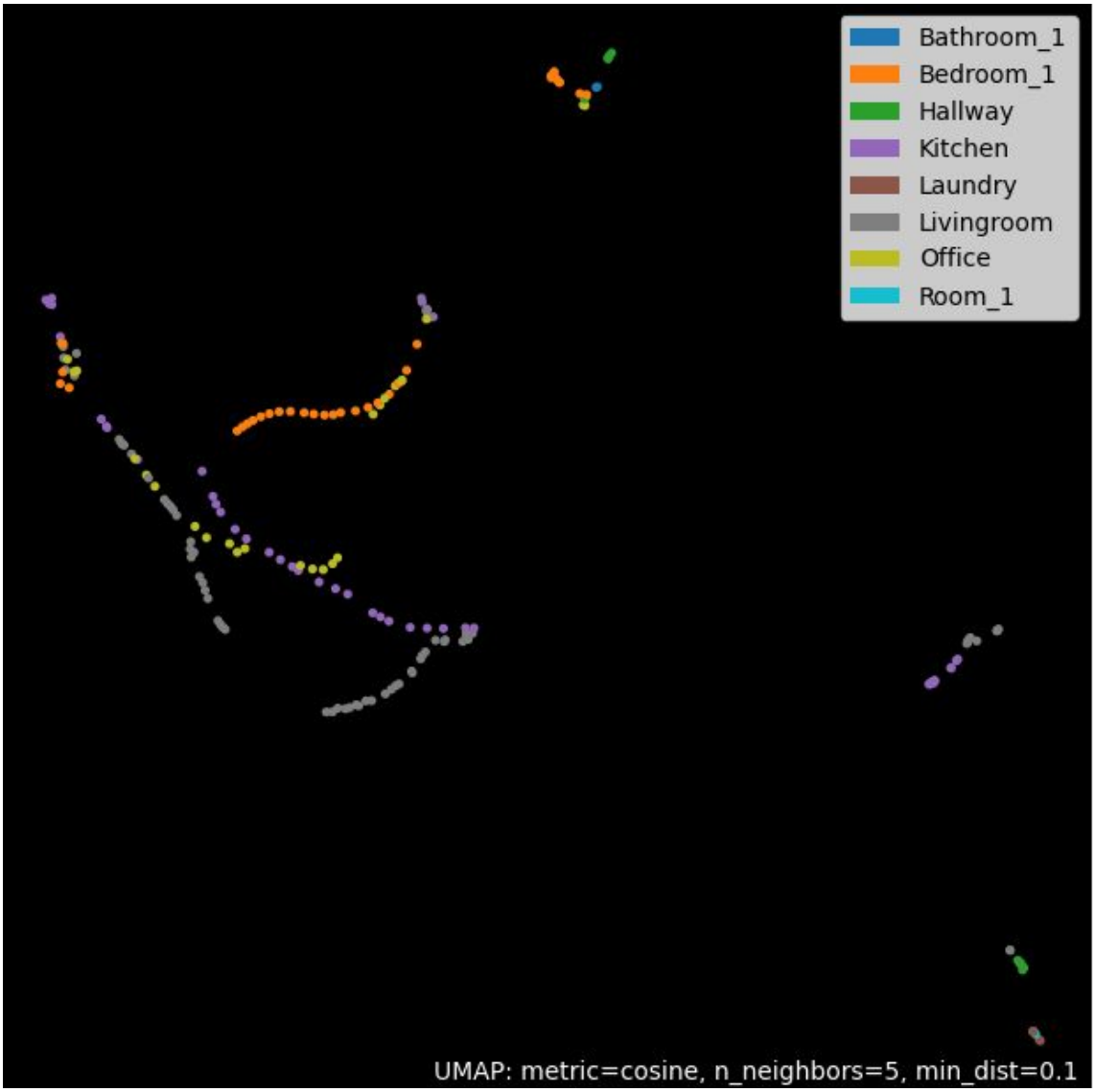
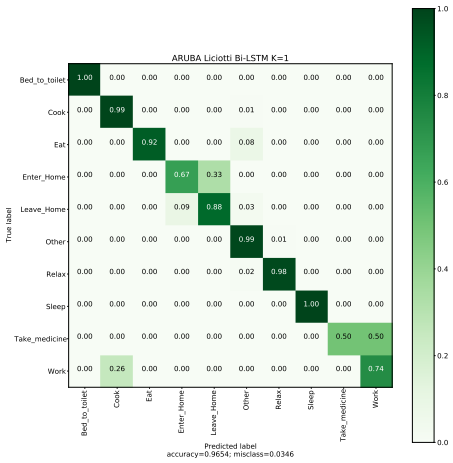


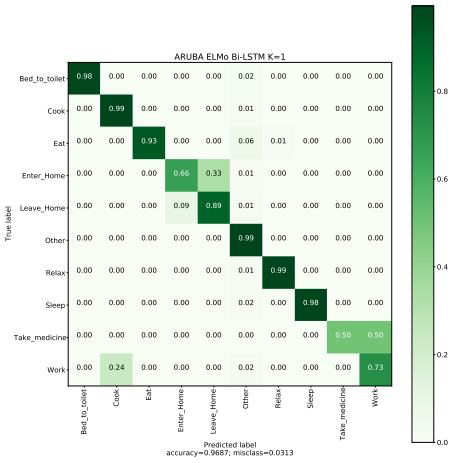
Figure B.4 – Cairo's cluster colorized by sensors' localisation

Appendix C

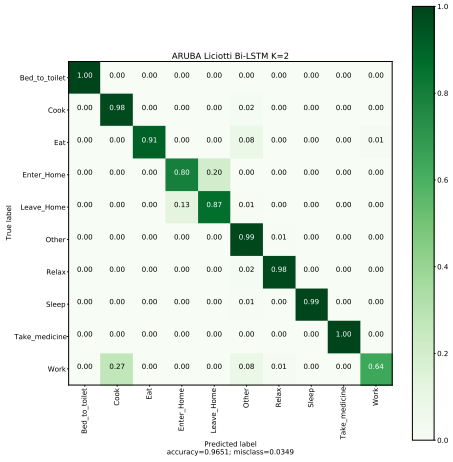
Experiments Confusion Matrices Grouped Activities



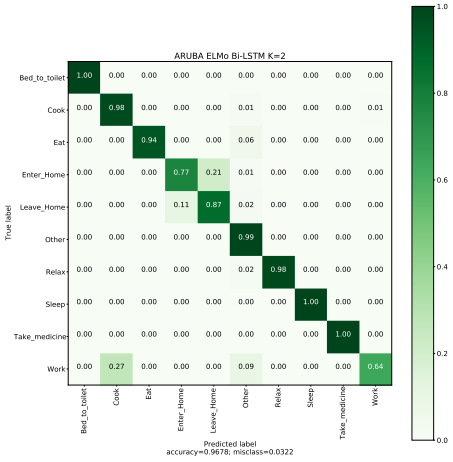
(a) Aruba Liciotti (Embedding + Bi-LSTM) K=1



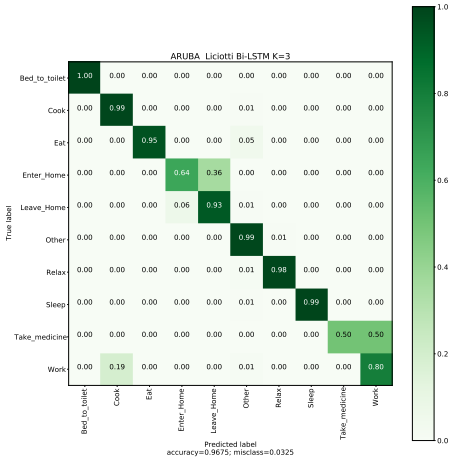
(b) Aruba ELMo + Bi-LSTM K=1



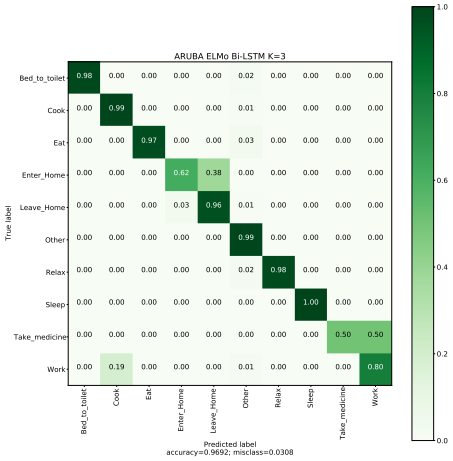
(c) Aruba Liciotti (Embedding + Bi-LSTM) K=2



(d) Aruba ELMo + Bi-LSTM K=2

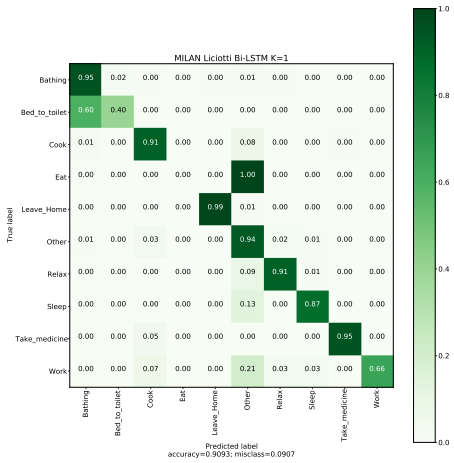


(e) Aruba Liciotti (Embedding + Bi-LSTM) K=3

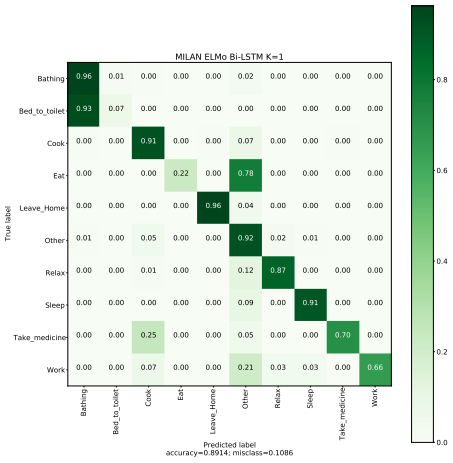


(f) Aruba ELMo + Bi-LSTM K=3

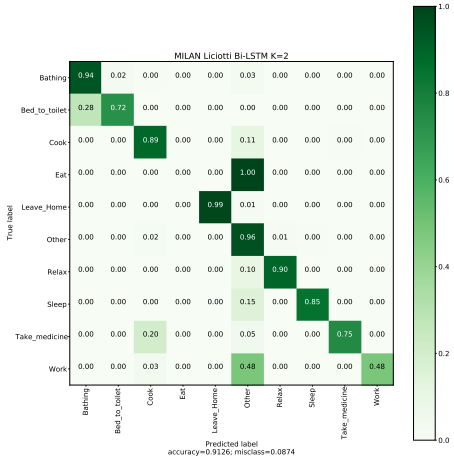
Figure C.1 – Aruba confusion matrices.



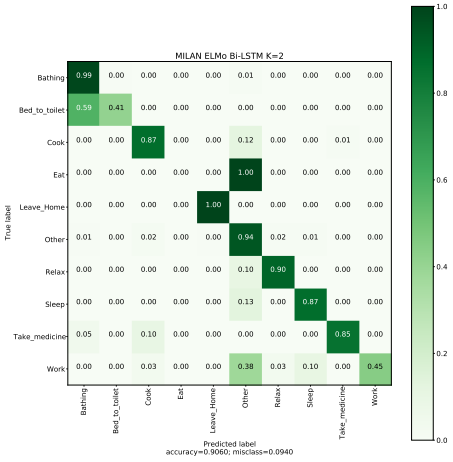
(a) Milan Liciotti (Embedding + Bi-LSTM) K=1



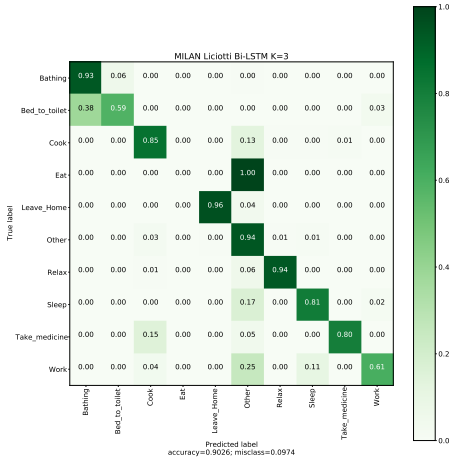
(b) Milan ELMo + Bi-LSTM K=1



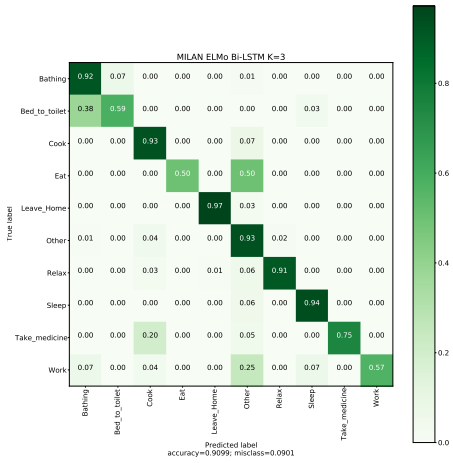
(c) Milan Liciotti (Embedding + Bi-LSTM) K=2



(d) Milan ELMo + Bi-LSTM K=2

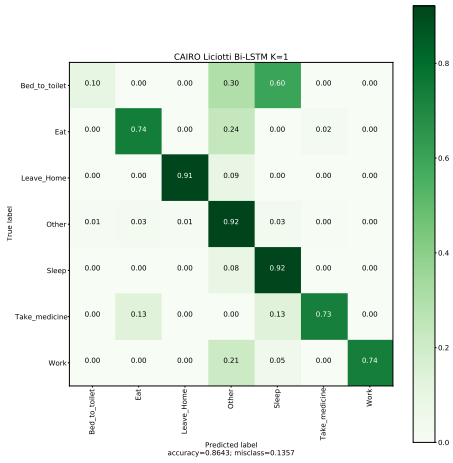


(e) Milan Liciotti (Embedding + Bi-LSTM) K=3

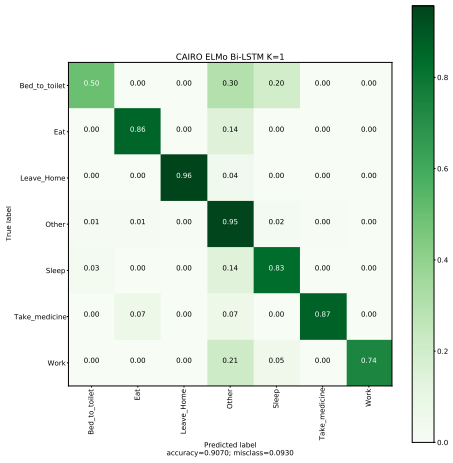


(f) Milan ELMo + Bi-LSTM K=3

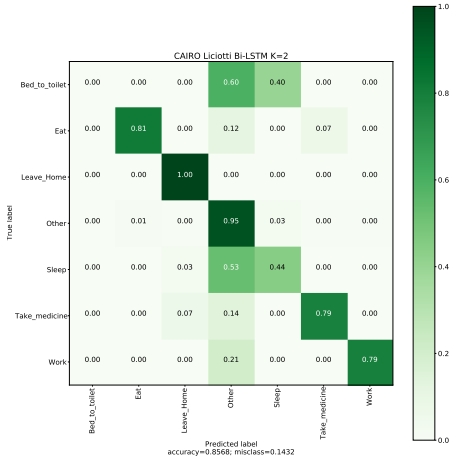
Figure C.2 – Milan confusion matrices.



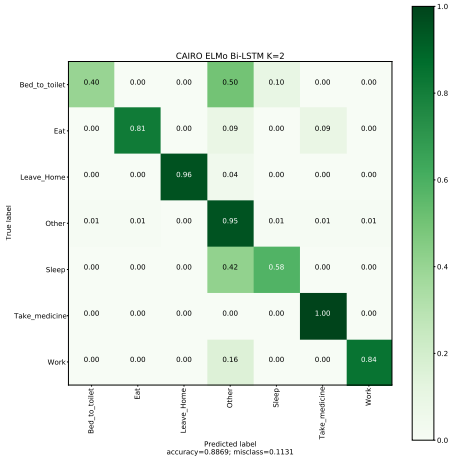
(a) Cairo Liciotti (Embedding + Bi-LSTM) K=1



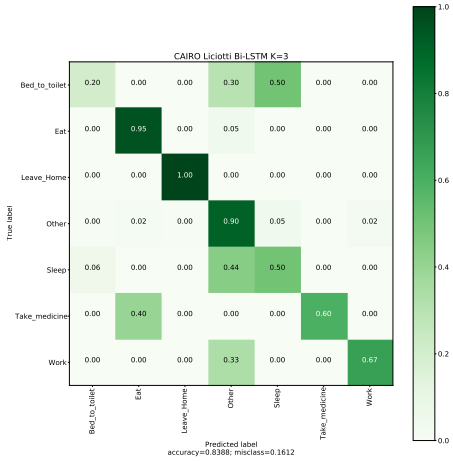
(b) Cairo ELMo + Bi-LSTM K=1



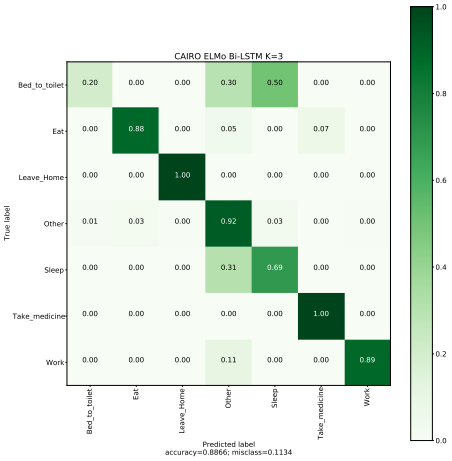
(c) Cairo Liciotti (Embedding + Bi-LSTM) K=2



(d) Cairo ELMo + Bi-LSTM K=2



(e) Cairo Liciotti (Embedding + Bi-LSTM) K=3

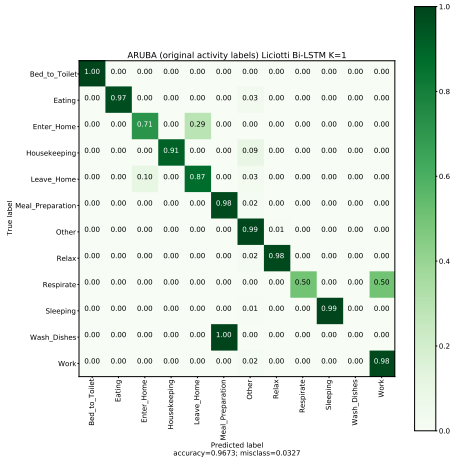


(f) Cairo ELMo + Bi-LSTM K=3

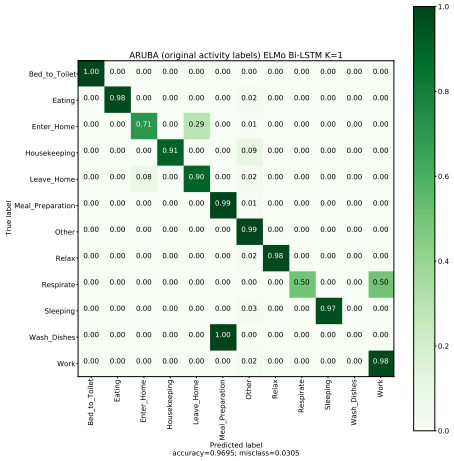
Figure C.3 – Cairo confusion matrices.

Appendix D

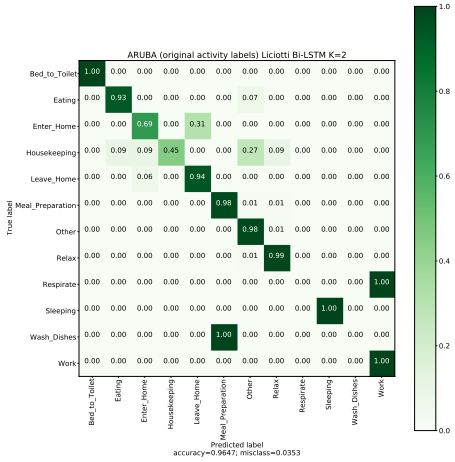
Experiments Confusion Matrices The Original Activities



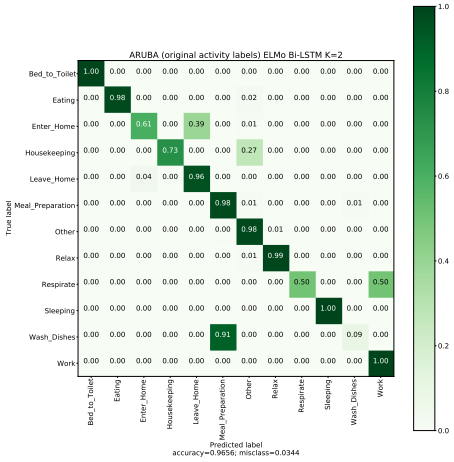
(a) Aruba Liciotti (Embedding + Bi-LSTM) K=1



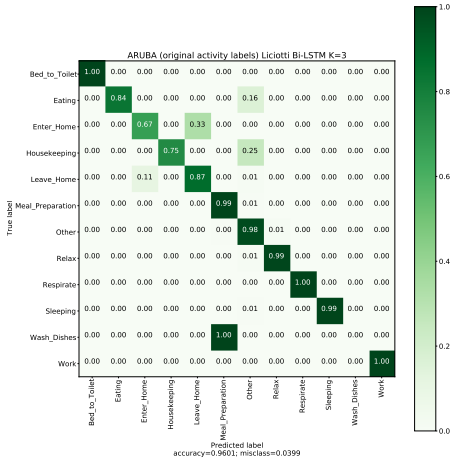
(b) Aruba ELMo + Bi-LSTM K=1



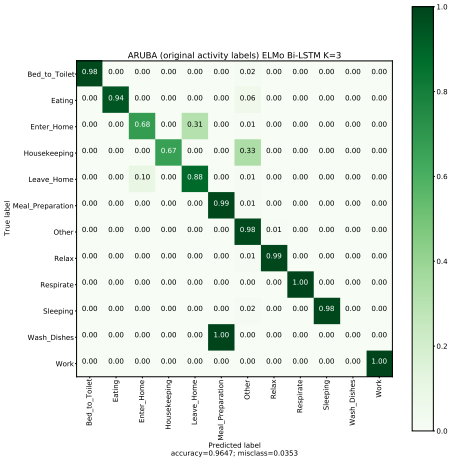
(c) Aruba Liciotti (Embedding + Bi-LSTM) K=2



(d) Aruba ELMo + Bi-LSTM K=2

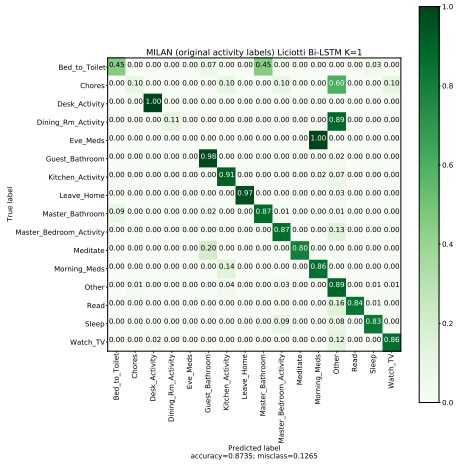


(e) Aruba Liciotti (Embedding + Bi-LSTM) K=3

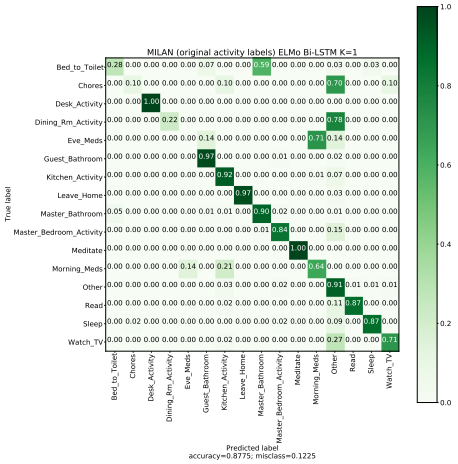


(f) Aruba ELMo + Bi-LSTM K=3

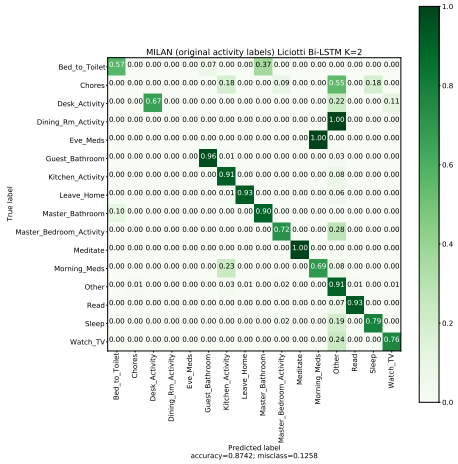
Figure D.1 – Aruba confusion matrices with the original activity labels



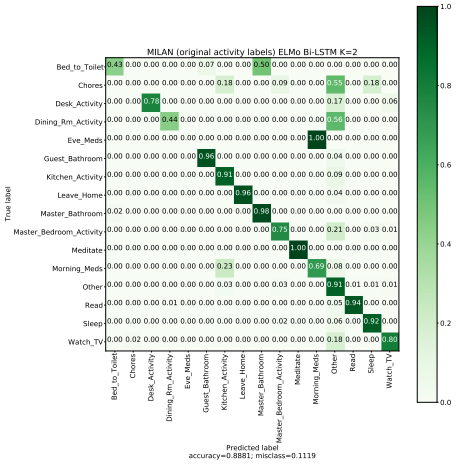
(a) Milan Liciotti (Embedding + Bi-LSTM) K=1



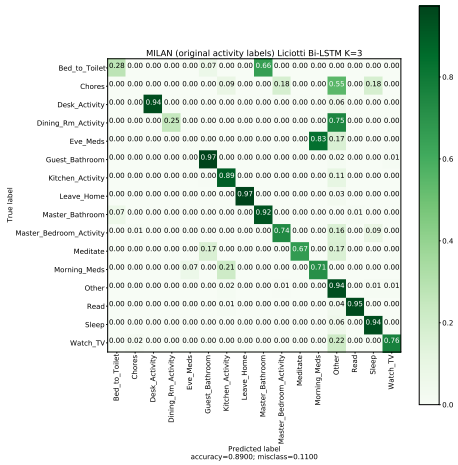
(b) Milan ELMo + Bi-LSTM K=1



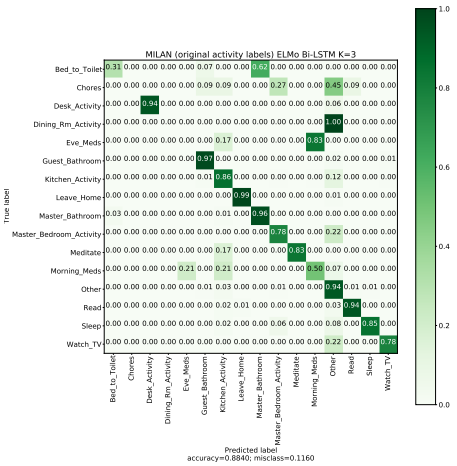
(c) Milan Liciotti (Embedding + Bi-LSTM) K=2



(d) Milan ELMo + Bi-LSTM K=2

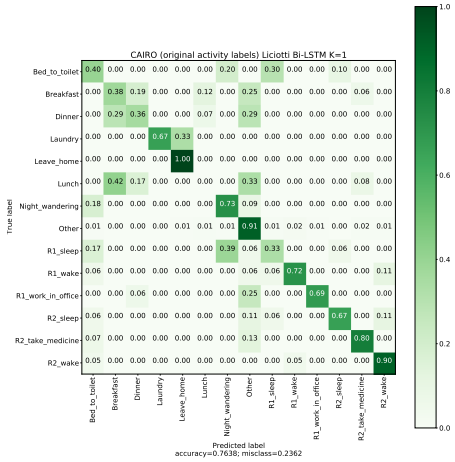


(e) Milan Liciotti (Embedding + Bi-LSTM) K=3

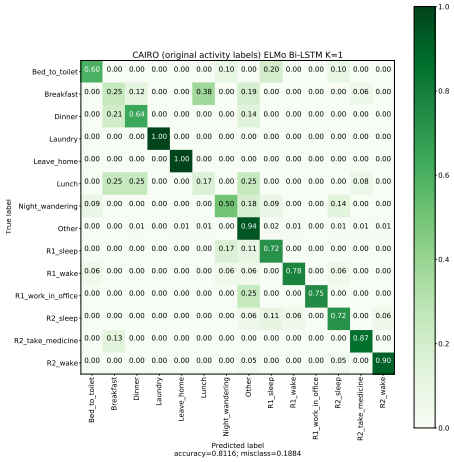


(f) Milan ELMo + Bi-LSTM K=3

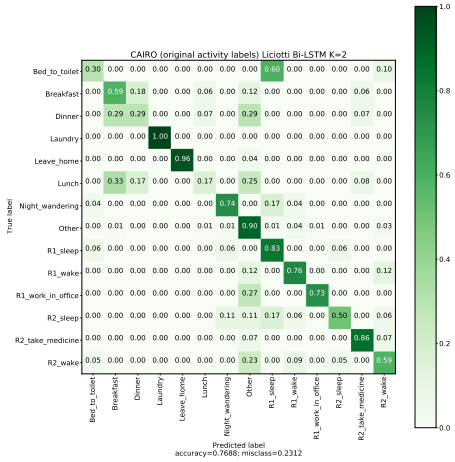
Figure D.2 – Milan confusion matrices with the original activity labels.



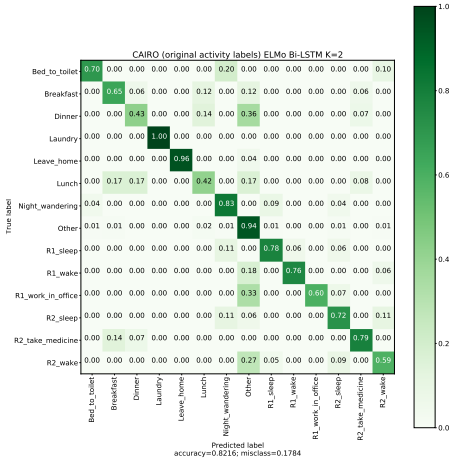
(a) Cairo Liciotti (Embedding + Bi-LSTM) K=1



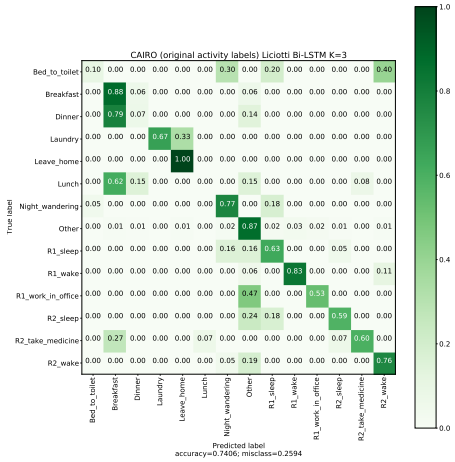
(b) Cairo ELMo + Bi-LSTM K=1



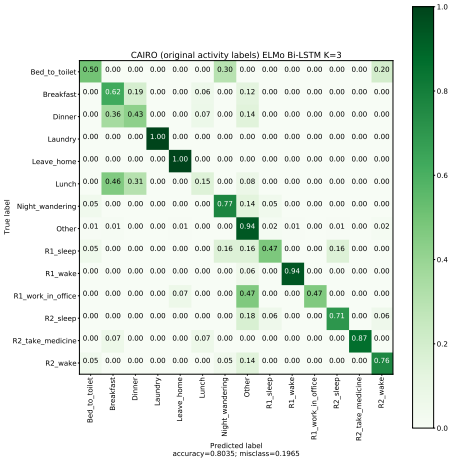
(c) Cairo Liciotti (Embedding + Bi-LSTM) K=2



(d) Cairo ELMo + Bi-LSTM K=2



(e) Cairo Liciotti (Embedding + Bi-LSTM) K=3



(f) Cairo ELMo + Bi-LSTM K=3

Figure D.3 – Cairo confusion matrices with the original activity labels.

Appendix E

Virtual Smart Home Configuration GUI

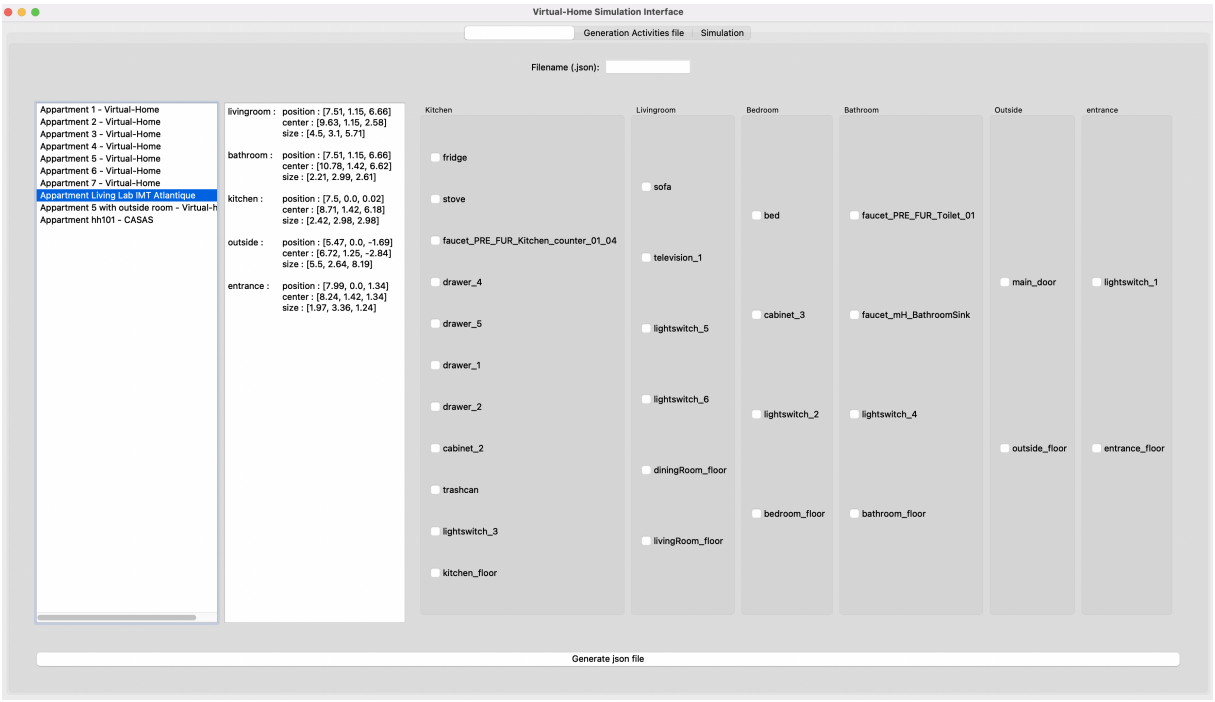


Figure E.1 – Scene Configurration

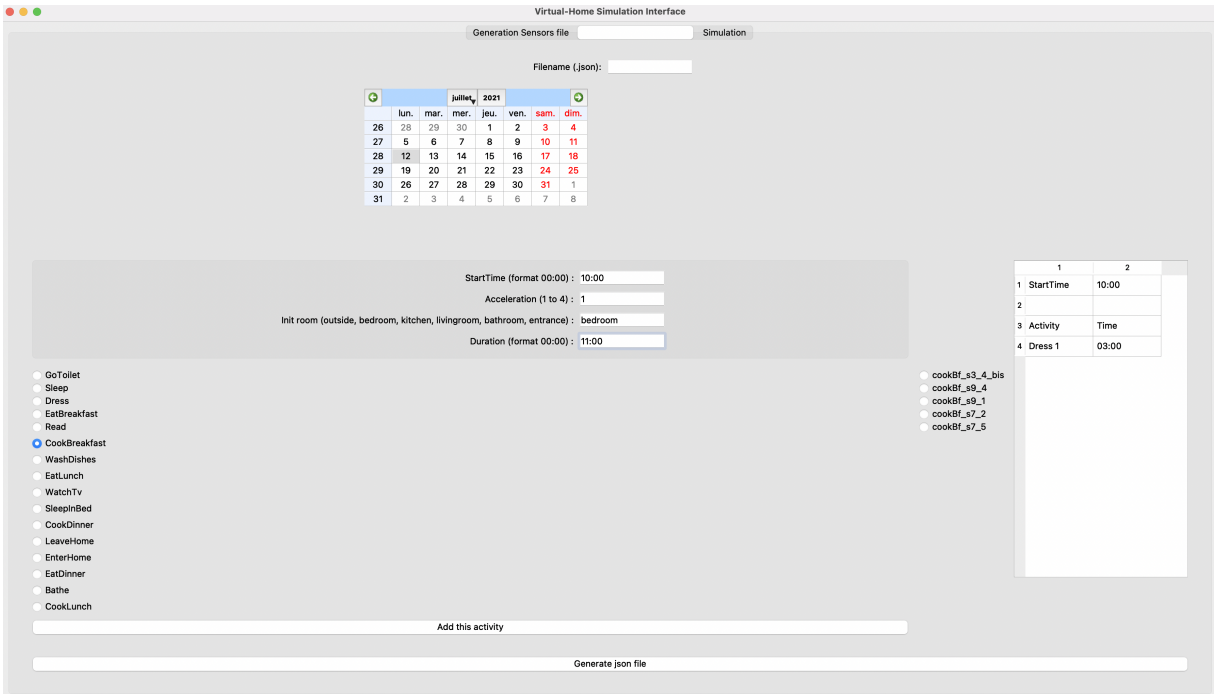


Figure E.2 – Scenario configuration

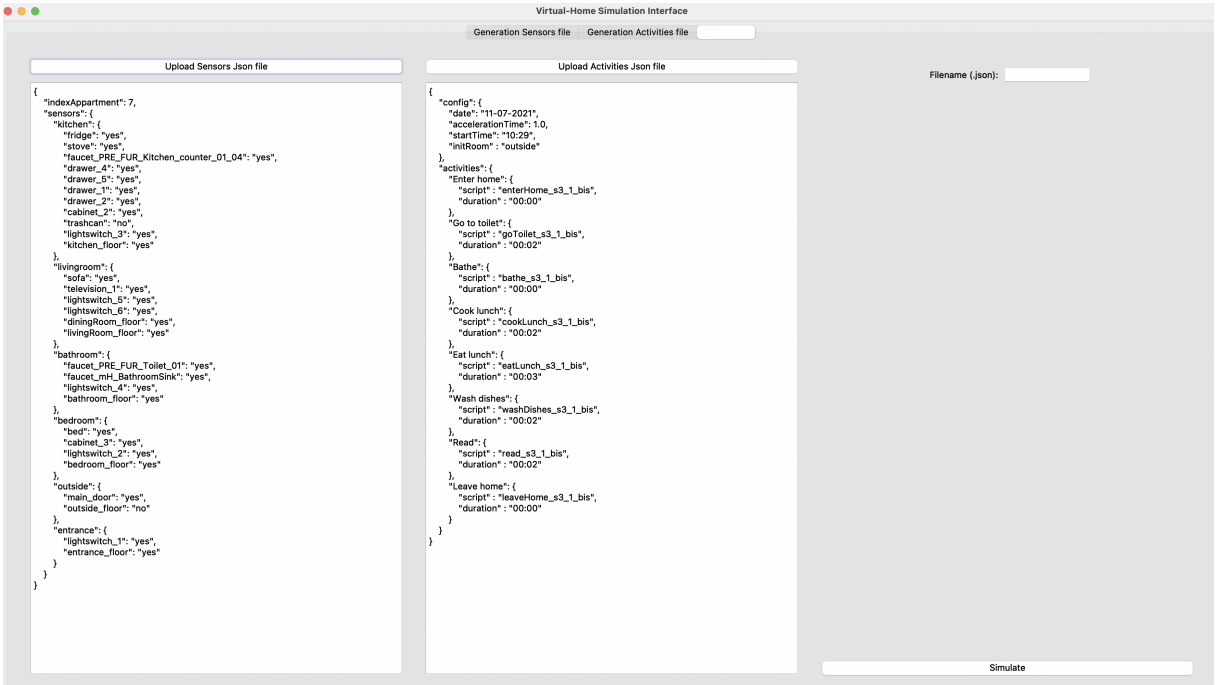


Figure E.3 – Check and start

DOCTORAT BRETAGNE LOIRE MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Reconnaissance de l'activité humaine dans les maisons intelligentes : traitement de la variabilité des données grâce à l'apprentissage contextuel, au transfert de connaissance et synthèse de données.

Mot clés : Reconnaissance de l'activité humaine, Maisons Intelligentes, Apprentissage Profond Contexte Dépendant, Transfert de Connaissance, Synthèse de Données, Jumeau Numérique

Résumé : La maison intelligente est au centre des attentions pour les nombreuses possibilités d'applications et de services innovants qu'elle peut offrir en termes de sécurité, d'économie d'énergie, d'amélioration du confort et d'aide à la santé. Ces dernières années ont vu émerger une multitude de techniques et d'approches au travers de l'intelligence artificielle pour doter les maisons de la pierre angulaire à tous ces services ; la capacité à comprendre le mode de vie et les activités de ses résidents au travers des capteurs domotiques. Malgré les avancées grâce au deep learning et l'augmentation de la quantité de données, ces méthodes ne permettent pas de généraliser et traiter totalement la complexité et la variabilité de l'activité humaine. De plus, l'utilisation et l'application de ces approches dans un contexte réel, industriel et commercial reste un défi majeur dû au manque de données labellisées en provenance de l'environnement de destination. Cette thèse propose de tenter d'améliorer ces méthodes et leurs portabilités au travers, de la compréhension du contexte d'activation des capteurs domotique et du transfert de connaissance en s'inspirant de techniques du Traitement du Langage Naturel, ainsi que de la synthèse de données via le concept du Digital Twin.

Title: Human Activity Recognition in Smart Homes: tackling data variability using context-dependent deep learning, transfer learning and data synthesis

Keywords: Human Activity Recognition, Smart Homes, Context-Dependent Deep Learning, Transfer Learning, Data Synthesis, Digital Twin

Abstract: The smart home is at the center of attention for the many innovative applications and services it can offer in terms of security, energy savings, comfort improvement and health support. The last few years have seen the emergence of a multitude of techniques and approaches through the artificial intelligence to bring the backbone of all these services to the home; the ability to understand the lifestyle and activities of its residents through home automation sensors. Despite advances in deep learning and the increase in the amount of data, these methods do not generalize and fully address the complexity and variability of human activity. Moreover, the use and the application of these approaches in a real, industrial and commercial context remains a major challenge due to the lack of labeled data from the destination environment. This thesis proposes to try to improve these methods and their portability through the understanding of the activation context of home automation sensors and the transfer of knowledge by drawing inspiration from Natural Language Processing techniques, and through data synthesis via the Digital Twin concept.