Biologically-Inspired Machine Intelligence Technique for Activity Classification in Smart Home Environments

Talal Sarheed Alshammari

A thesis submitted in partial fulfillment of the requirement of Staffordshire University for the degree of Doctor of Philosophy

May 2019

List of Publications

- **REFEREED JOURNAL PUBLICATIONS**

OpenSHS: Open Smart Home Simulator **May, 2017**
Sensors Journal MDPI

- **REFEREED JOURNAL PUBLICATIONS**

SIMADL: Simulated Activities of Daily Living Dataset. **April, 2018**
Data Journal MDPI

- **REFEREED CONFERENCE PUBLICATIONS**

Evaluating Machine Learning Techniques for Activity Classification in Smart Home Environments **February ,2018**
World Academy of Science, Engineering and Technology International Journal of Information and Communication Engineering

Acknowledgements

I would like to thank my supervisors Dr. Mohamed Sedky and Chris Howard for encouraging and supporting me during my Ph.D study. They were very helpful and provided me their assistance throughout my Ph.D journey. It was very good experience to improve my research skills. Due to this experience, I learned new knowledge, found new ideas from your supervision. I appreciate your consideration, efforts and kind cooperation, you deserve all the thanks.

I would like to thank to my parents and my wife for their unconditional support that has helped me for achieving this thesis.

I would like to thank the Ministry of Education in Saudi Arabia and Hail university for funding and supporting this research.

Contents

Abstract

With the widespread adoption of Internet-connected devices and the prevalence of applications in the Internet of things (IoT), devices in smart homes can generate enormous amounts of data. There is a requirement for machine-learning techniques to learn from historical patterns and predict future activities. There is an increased interest in machine-learning techniques that can provide useful and interesting services in the smart home domain. The areas that machine-learning techniques can help advance are varied and ever-evolving. Predicting and classifying the Activities of Daily Living (ADLs) of inhabitants in a smart home environment are key modules to automate smart home devices. Some prominent examples include uses for entertainment, elderly care, healthcare, and security. The abilities of the machine-learning technique to find meaningful spatio-temporal relations of high-dimensional data and to learn from streaming datasets are important requirements.

Recently, the Hierarchical Temporal Memory (HTM) theory has been presented as a biologically-inspired machine-learning theory that attempts to mimic the neocortex, the front part of the human brain. The main features of the HTM theory include the ability to learn and predict temporal patterns. The HTM theory and its computational implementation, Cortical Learning Algorithms (CLA), present a potential alternative to traditional machine intelligence. This research aims to apply a new biologically inspired machine intelligence technique based on the HTM theory and its CLA implementation to classify ADLs of inhabitants by analysing data captured from different sensors in a smart home scenario.

This research started by reviewing existing research in classification and prediction of ADLs. A comprehensive evaluation of state-of-the-art machine learning techniques and their application in the context of smart homes has been carried out as a primary research. HTM theory and its implementation the CLA have been studied, and experiments were conducted to identify the weaknesses and limitations of applying the HTM theory and the CLA for activity classification in the smart home environment.

To test and evaluate the performance of a proposed machine intelligence technique, there is a need for a dataset that represents the ADLs in a smart home scenario. Due to the excessive cost of building real smart home datasets and the lack of real datasets from smart homes, to tackle this issue, as a secondary contribution to knowledge, this research used OpenSHS (Open Smart Home Simulator), an open-source, cross-platform 3D smart home simulator. In addition, forty-two ADL datasets, Simulated Activities of Daily Living Dataset (SIMADL) were extracted from the OpenSHS tool and made available publicly.

This research proposes multi-region CLA techniques to learn short- and long-term patterns. Two novel multi-region CLA techniques featuring a multiple spatial pooler and temporal memory regions that incorporate a hash encoder and a Multi-Layer Perceptron (MLP) classifier were proposed and applied. While, the hash encoder can deal with multi-dimensional datasets, because of the existing encoders of the standard NuPIC encoders are prepared to deal with a single column or a small number of columns and the MLP classifier was used rather than the classifiers used in the current implementation of CLA to produce meaningful predictions. Additionally, the two novel multi-region CLAs, Parallel Spatio-Temporal Memory Stream (CLA2) and Cascaded Temporal Memories Stream (CLA3) were developed to learn short- and long-term patterns from, streaming datasets. To remove the limitation of memory management of the original CLA that contains one spatial pooler and one temporal memory, the original CLA learns using one memory level, such model learns either short-term or long-term patterns, not both of them. A novel CLA2 was proposed and developed, to cope with learning both short-term and long-term patterns. CLA2 can learn both short-term and long-term patterns in parallel. CLA2 includes two spatial poolers and two temporal memories to simulate short-term and long-term memories. The number of cells per column was decreased in the first region to learn short-term patterns. In the second region, the number of cells per column was increased to learn long-term patterns and the outputs of both regions were concatenated into one vector. The second proposed algorithm (CLA3) has three regions, one spatial pooler, and three cascaded temporal Memories (TM) regions, where the first region learns smaller features, and the second and third regions are more abstract in order to learn and recognise patterns and concatenates the outputs from all three regions into one vector.

An evaluation and comparison of the proposed algorithms against state-of-the-art supervised machine-learning techniques and the standard CLA for classification was conducted using both the simulated smart home SIMADL dataset, generated using the OpenSHS, and the ARAS dataset, that comprises data captured from real-world activities of residents residing in two houses, the obtained results for the real datasets offered less performance than the synthetic datasets. Because the inhabitants are asked to record their activities manually, it was prone to human errors. The real-world dataset ARAS House A is inconsistent because one of the inhabitants left the house for long period of time, which impacts the performance results. The results of the proposed algorithms for the classification of ADLs show that its performance are promising. For the CLA2, the average overall F-measure for all the synthetic datasets, SIMADL and the real-world datasets, ARAS is 84.88%, while the highest F-measure (86.87%) was achieved by the Convolutional Neural Network (CNN) model. The (CLA2) has achieved an F-measure of 92.63% for House B of the real-world ARAS dataset, which outperforms state-of-the-art classification models. For the CLA3, the average overall F-measure for all the synthetic and real datasets is 84.50% . The proposed algorithms improve the best performance of base-line (standard) CLAs by an average F-measure of 51.81% overall.

List of Figures

List of Tables

Abbreviations

ADL	**A**ctivities of **D**aily **L**iving
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etworks
CLA	**C**ortical **L**earning **A**lgorithm
DT	**D**ecision **T**rees
HMM	**H**idden **M**arkov **M**odel
HTM	**H**ierarchical **T**emporal **M**emory
IoT	**I**nternet **o**f **T**hings
KNN	**K** **N**earest **N**eighbour
MLP	**M**ulti **L**ayer **P**erceptron
NuPIC	**Nu**menta **P**latform for **I**ntelligent **C**omputing
LSTM	**L**ong **S**hort **T**erm **M**emory
OpenSHS	**O**pen **S**mart **H**ome **S**imulator
PSO	**P**article **S**warm **O**ptimisation
RFID	**R**adio **F**requency **ID**entification
SDR	**S**parse **D**istributed **R**epresentations
SIMADL	**S**imulated **A**ctivities of **D**aily **L**iving **D**ataset
SHE	**S**mart **H**ome **E**nvironments
SP	**S**patial **P**ooler
SGD	**S**tochastic **G**radient **D**escent
SVM	**S**upport **V**ector **M**achines
TM	**T**emporal **M**emory

Chapter 1

Introduction

1.1 Problem Definition

Existing homes consist of many systems for energy, lighting, heating, cooling, etc. that could generate enormous amounts of data. This collective of devices and the data produced can be sent to the Internet to form what is called the Internet of Things (IoT). Stakeholders such as homeowners, city officials, businesses and citizens need to analyse this generated data to make timely decisions. Learning from historical events and attempting to predict future events is an essential requirement for the future home. Homeowners require a technique that allows them to receive predictions of the consequences of their activities to inform interventions. They also would like to measure the success of their interventions. Machine intelligence techniques have the potential to help homeowners to analyse the data gathered from their homes and to predict future activities within their homes. Although several attempts have been made to use machine intelligence techniques to make our homes smarter, however, such methods are not designed to learn from a stream of data, hence there is a need to investigate alternatives. Recently, Hierarchical Temporal Memory (HTM) theory has been presented as a bio-inspired machine learning theory that attempts to mimic the Neocortex, the front part of a human brain. HTM main features include the ability to learn and predict temporal patterns. Although, HTM and its computational implementation, Cortical Learning Algorithms (CLA), presents a potential alternative to traditional machine intelligence, however, this new technology has not been fully investigated in the context of smart homes, apart from work proposed by (Otahal & Stepankova, 2014). Moreover, up to the knowledge of the researcher, the literature does not include a comprehensive study that identifies the limitations and weaknesses of applying CLA in smart homes.

1.2 Aim

The aim of this research is to propose a new biologically-inspired machine intelligence technique to classify and predict ADLs of inhabitants, within a smart home environment, and to evaluate its performance. The HTM theory will be used as the base for the proposed technique. The first phase of this project includes the application of HTMs implementation, the CLA, in a smart home environment. Then, the evaluation of the performance of the CLA in smart home scenarios will take place to identify its weaknesses and limitations. In the second phase, the CLA will be modified to improve its performance in order to meet the requirements of the application at hand.

1.3 Objectives

The objectives of this project are to:

1. Carry out a literature review covering related work on the Internet of Things (IoT), its major applications, requirements and challenges and more specifically on smart homes,
2. Carry out a literature review on the use of machine intelligence techniques for prediction and classification,
3. Analyse the literature review to evaluate the requirements for machine intelligence techniques to classify events within the context of smart homes,
4. Study HTM theory and its implementation, the CLA, including its learning and prediction capabilities, investigate the use of (HTM) theory as effective classification algorithms for smart homes,
5. Apply the CLA in the context of smart home,
6. Use a smart home simulator and extract relevant datasets, that simulate the smart home environment, to test the proposed algorithm, using a dataset,
7. Evaluate the performance of the CLA to identify its weaknesses and limitations,
8. Propose a new algorithm to improve CLA performance in the smart home domain,
9. Evaluate the test results and assess the proposed algorithm against state-of-the-art machine learning techniques.

1.4 Research Questions

- What are the weaknesses and limitations of applying the HTM theory in smart home domain? And, how to overcome these limitations?
- To what extent is the HTM theory suitable for classification of ADLs in smart home domain? And, how can we exploit the potential of HTM in this application?

1.5 Scope of the Research

This research implements a classification algorithm based on HTM theory and its CLA implementation. The proposed algorithm attempts to classify ADL of inhabitant's that are captured from different binary sensors in a smart home scenario.

IoT paradigm contains several applications, for example, smart home, smart cities, smart building, etc. The smart home is one of the most important parts of the IoT paradigm. Smart home consists of Passive, Reactive, Proactive statements or behaviours, each sensor generates enormous amounts of data. There is a need for a machine learning algorithm that has a level of intelligence that is able to perform modelling, reasoning and integration such vast amount of data. Machine learning techniques could be divided into three categories based on their learning type: supervised learning, unsupervised learning and reinforcement learning. Each technique use its own probabilistic, statistical methods and sequence-learning algorithms, Figure 1.1 shows the scope of this research.

The smart home is a small-scale example of a smart city, where city officials need to link and correlate data combined from various sources and make sense of the changes in such data over time. However, the data of a smart city are difficult to find. The smart-home domain has been chosen as the case study for this project because smart home datasets are available in the public domain.
The machine learning algorithms covered in Figure 1.1 are not application-specific and have been used across different application areas. Although, this research has chosen the smart home domain as a case study representative of an IoT application area. However, the machine learning algorithms shown in Figure 1.1 are capable of scaling up to other IoT application areas but further investigation is required to empirically test them in the other domains.

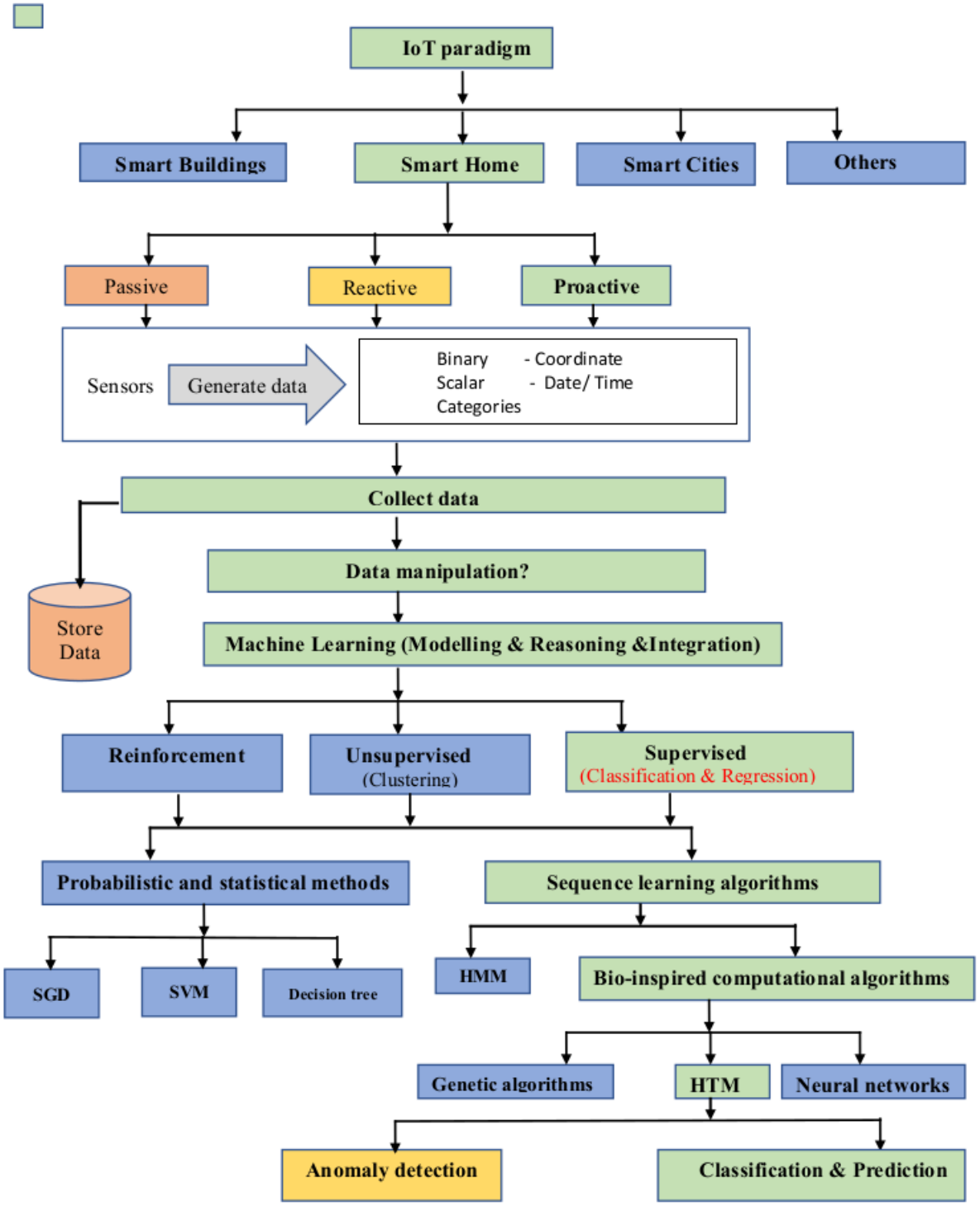


FIGURE 1.1: The Scope of This Research.

1.6 Challenges

These challenges include the design of an efficient machine learning algorithm that can classify ADLs, the evaluation of different machine learning algorithms in this context

which requires a suitable dataset that represents different smart home scenarios.

- Activity Prediction
  The prediction of the inhabitants activities in a smart home environment is a challenging task. This is because a smart home requires certain types of intelligence, such as adapting to the users changing habits, being able to deal with streaming data and coping with noisy data such as missing data and faulty sensors data. (Minor *et al.*, 2015), (Fatima *et al.*, 2012) summarises the challenges of activity prediction and classification in a smart home as:

  1.) the data generated in a smart home includes multi-dimensional features, hence, the activity recognition in a smart home domain requires a multi-class classification method which is prone to errors,
  2.) the data readings from sensors are always noisy which makes it difficult to exploit the relationships between spatial and temporal attributes of these readings..

- Performance Evaluation
  Although there are many technological advances in the field of IoT, there are equally as many obstacles related to the implementation and evaluation of newly proposed IoT technologies. This is particularly seen when addressing practical or realistic conditions for real world deployments (Sanchez *et al.*, 2014). Evaluating the algorithm's performance for activity classification through metrics is challenging because some algorithms requires parameter tuning for each task that affects the algorithm's performance (Minor *et al.*, 2015).

- Smart Home Dataset Generation

  The development, testing, and evaluation of machine learning algorithms and pervasive computing requires suitable datasets. The evaluation of such machine learning algorithms is further limited by the lack of real datasets from smart homes. Due to the limitations and high cost of building real smart home datasets, there is a need for powerful simulation tools that can represent the home inhabitant's daily activities. These simulation tools can generate datasets with higher flexibility (Cook *et al.*, 2009),(Bouchard *et al.*, 2010), (Synnott *et al.*, 2015).

1.7 Main Contributions

The main contributions to knowledge of this research project, ordered by importance, include the following:

- Two novel techniques that learn user patterns and allow the home user to obtain predictions based on the HTM theory and CLA. This is envisaged to be achieved by proposing two novel multi-region CLA techniques that include a hash encoder and an MLP classifier.
- The forty-two SIMADL datasets are generated using simulation tool (OpenSHS) for ADL classification problems that simulates a smart home environment.
- A review of existing research in classification and prediction of ADLs and carries out a comprehensive evaluation of state-of-the-art machine-learning technique applications in the context of smart homes.

1.8 Research Methodology

Researchers must adapt a strategy to answer their research questions. Research methods include two common research approaches: qualitative and quantitative. The choice of method depends on the type of research (Brannen, 2005).

1.8.1 Quantitative Research

Quantitative research is the process of data collection, analysis, and interpretation to obtain the results of a study. It is used by investigators to monitor and measure phenomena via statistical and mathematical methods. Quantitative research defines a numerical or statistical design approach to research. Quantitative research builds on current theories, making it specific in its surveying and experimentation (Williams, 2007).

1.8.2 Qualitative Research

Qualitative research is a method that involves discovery, such as an exploration paradigm that happens in natural environments, which leads investigators to develop a level of knowledge by sharing current experiences. One identifier of qualitative research is a

social phenomenon explored from the researchers perspective. There are many new issues and subjects that can be discovered through the opinions of certain people. It creates non-numerical data, using methods such as case studies, ethnographic studies, and interviews (Williams, 2007).

Due to the nature of the data and the use of machine-intelligence techniques based on the HTM theory and its CLA implementation to classify activities of daily living (ADLs) of inhabitants in a smart home, this current research adopts a quantitative research approach in the design, implementation, testing, and evaluation stages. This research uses a 3D simulation tool (OpenSHS) to simulate a smart-home environment for ADL classification problems, where data can be generated and stored to create a dataset. The datasets can be used to evaluate and test the performance of machine-learning models.

The hypothesis is that HTM and its CLA are able to integrate and learn patterns from streaming datasets to classify activities. The dataset is used with state-of-the-art machine-learning algorithms to compare the performance with the proposed techniques. In the evaluation stage, the system output is quantitatively measured to evaluate its effectiveness through metrics, depending on the evaluation criteria. As shown in Figure 1.2 the research methodology, objectives, methods are presented.

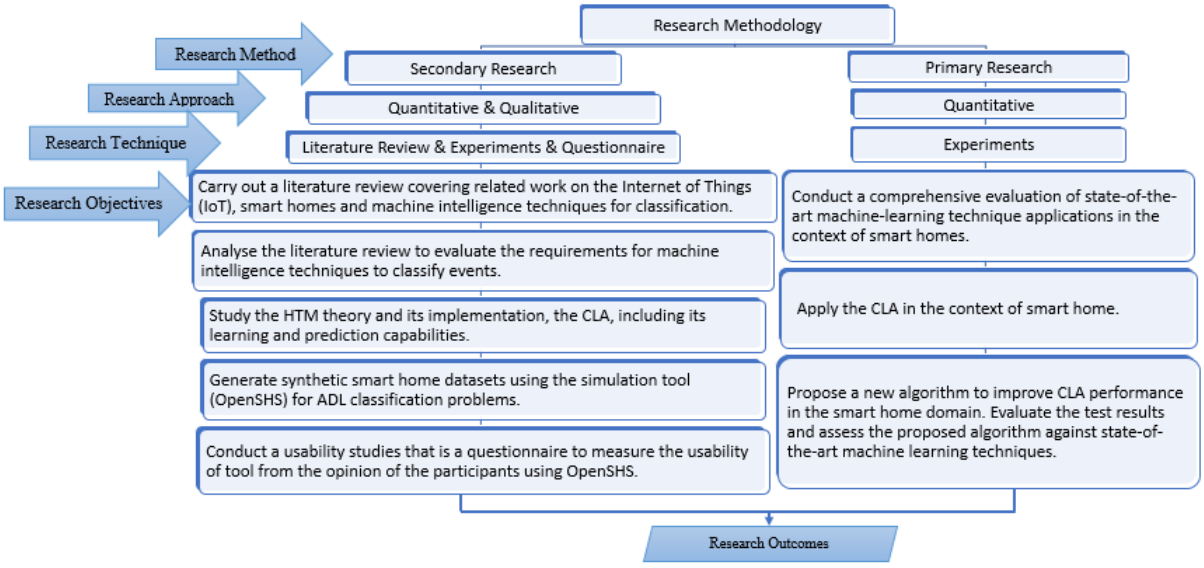


FIGURE 1.2: Research methodology, objectives and methods.

1.9 Organisation of the Thesis

This thesis is structured as follows:

- Chapter 1 covers the problem definition, project's aim, objectives, scope of the research, research questions, challenges and main contributions.
- Chapter 2 reviews the IoT literature and focuses on smart home applications and machine intelligence algorithms for the smart home covering, middleware, intelligence, context-awareness, related work, analyses and research the gap.
- Chapter 3 introduces the concept of the Hierarchical Temporal Memory (HTM) theory and its components, Sparse Distributed Representation (SDR), encoder, spatial pooler, temporal memory, CLAClassifier as well as the implementation of the theory, Cortical Learning Algorithm (CLA).
- Chapter 4 introduces the Smart Home SIMADL Datasets generated by OpenSHS for classification problems. This chapter is structured as follows: Section 2 presents the related real-world datasets and simulation tools in the literature. Section 3 explains OpenSHS architecture and how this research uses it to generate the SIMADL datasets. Section 4 presents the methodology to generate the SIMADL datasets. Section 5 provides a description of the SIMADL datasets, as well Usability study.
- Chapter 5, First section: introduces the Hash SDR Encoder for Smart Home. Explains the Hash Encoder, problem definition for the Hash encoder, how does the hash function work. Second section: introduces the classifier. Explains Multi-Layer Perceptron, Backpropagation algorithm and how the application of the MLP classifiers works. Third section: introduces the multi region HTM, offers biological short and long-term memory, illustrates the capacity of sequential memory to represent different forms, reviews the related work multi-region HTM in the literature, Explains the proposes multi-region CLA techniques - Parallel Spatio-Temporal Memory Stream and Cascaded Temporal Memories Stream, how the proposed model works and analysis both of them.
- Chapter 6 includes two parts; the first part covers evaluation methodologies, the second part describes the preparation of the real-world datasets, ARAS, the synthetic SIMADL datasets, parameter selection for the CLA model, evaluates the proposed algorithms against state-of-the-art supervised machine-learning techniques and and base-line (standard)CLAs, presents the test results.
- Chapter 7 concludes the thesis and includes a summary, future work, as well as, the original contributions.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the IoT literature including the major issues, enabling technologies, middleware, context-awareness and focuses on smart home applications and machine intelligence algorithms for the smart home covering, intelligence, related work, analyses and research the gap.

2.2 Internet of Things (IoT)

The first use of the term IoT was in the early twenty-first century, in 1999, by Kevin Ashton, regarding his idea to link some digital devices that exist around us in a household in a way that allows us to know their status remotely (Ashton *et al.*, 2009). The IoT is a sophisticated concept for the Internet that proposes that all of the things in our lives could be connected to the Internet or could be connected with each other to send and receive data to perform specific functions through the network. Recently, the IoT has emerged as an umbrella that covers different technologies concerning the connection of our things (devices) using the Internet. The IoT attempts to link distributed devices and sensors by using information technology, this enables new classes of applications and services. An enabled physical object is easily integrated into a network to become active in business, commerce or other services. Although the obvious benefits from the IoT, however, there are a number of issues that hinder the advances of IoT applications for example, the automation, standardization, security and privacy issues (Miorandi *et al.*, 2012), (Haller *et al.*, 2009). Automation is a challenging problem for developing automated systems to control devices in the home. (Youngblood *et al.*, 2005a) presented agent architecture in

order to automate a smart home that used techniques to decrease residents interactions by 72.2%. (Cook *et al.*, 2003) designed an intelligent environment called MavHome that includes prediction algorithms to predict the inhabitant's next action to automate repetitive tasks for the residents. Standardisation, heterogeneity, and interoperability between devices in the Internet of things (IoT) are challenging problems. They require developing a multifaceted technology approach to the IoT in order to exchange information and communicate between two or more devices (Elkhodr *et al.*, 2016). Google discussed a new networking protocol in order to find a standard for connection among devices in a home. However, there are many companies establishing standards for the IoT, such as Intel, Qualcomm, and GE (Neagle, 2014). There are security and privacy threats when using it. Attacks must be intercepted, data should be authenticated, access controlled and the privacy of customers need to be considered (Weber, 2010). Sarma and Giro concluded that there are still two issues, at least until the research community finds an optimal IoT solution in the future: preserving security and putting the user back in control and moving to the IoT. They proposed a solution to solve issues related to the diversification of the Internet to an IoT with new accessibility methods in the digital world. Their research proposes two approaches to deal with these issues: The Identinet approach, where identities are at the end point of all communications, and a concept designated by a digital shadow. The Identinet can represent an entity that includes devices, people and software. Digital shadowing presents a concept of entities which use nodes, services and equipment infrastructure in individual contexts. This can help users to attach to multiple entry points which keep a consistent view (Sarma & Girão, 2009).

2.2.1 Major Issues

There are many obstacles facing the IoT vision such as standardisation, fragmentation and interoperability. (Bandyopadhyay & Sen, 2011) identified some major issues in IoT. They were seeking to achieve complete interoperability among connected devices and support these devices to become more intelligent and be adaptive to the changing habits of the user. The standardisation is another major issue facing the widespread of architectures, identification schemes and protocols to be working together. The lack of standardisation leads to fragmentation of the IoT. (Atzori *et al.*, 2010) conducted a comprehensive survey about the IoT. In a system which contains massive numbers of nodes, the number of required IPv4 addresses to identify each node is limited and it is not enough. Thus, in order to solve this issue, IPv6 addresses will be used for low-power wireless communication nodes within Low-power Wireless Personal Area Networks (6LOWPAN) context and is enough to identify any object.

2.2.2 Enabling Technologies

The realisation of the IoT vision was facilitated by using sophisticated communication technologies such as Radio-Frequency Identification (RFID), Wireless Sensor Networks (WSN) and 6LoWPAN. (Srivastava & Saxena, 2006) presented a study that wireless technologies play an important role among radios and humans. (Juels, 2006) RFID is used in order to observe objects in real-time. It is not needed to be in line-of-sight, therefore it allows for mapping the real world into the virtual world. (Welbourne *et al.*, 2009) created a mini IoT network utilising the RFID Ecosystem in order to enable users to control their personal RFID data, they carried out a study for one month to assess trends in adoption using the applications and the users' reactions. Fosstrak[1] is a project that concentrates on the management of RFID relevant applications, It is an open source RFID software platform that implements the EPC Network specifications. (Gubbi *et al.*, 2013) identified many technologies including the RFID and WSN as enabling technologies that are probable to drive IoT research in the future.

2.2.3 Middleware

The IoT middleware is a software that serves as an interface between different IoT components, a networked operating system and smart home applications. The middleware is used to overcome issues, such as integrating heterogeneous features, security and interoperability (Issarny *et al.*, 2007). (Kawsar, 2009) and (Kawsar & Nakajima, 2009) offered a document-centric framework for constructing distributed smart object systems. There are many applications for smart objects that can be presented as a group of tasks in a document to solve the heterogeneity issue of smart objects. Infrastructure can administer the smart objects; hence, application development will be fast and straightforward. The framework provided by the programming abstraction enables the application developers to expand the tasks of smart objects; hence, it can improve many of the functionalities for smart objects where there are available alternatives to achieve the needs and requirements of the IoT vision. (Bandyopadhyay *et al.*, 2011) presented a study about the existing middleware in the IoT field. IoT requires cooperation with heterogeneous, device management, interoperation, context awareness, security and privacy, and managing data volumes. Especially context awareness, it must be within the IoT middleware to work in a smart home. The study illustrated a lack in context awareness; hence, it needs to understand the functions in the existing IoT middleware to improve it.

[1] http://www.fosstrak.org/(accessed February-2019)

2.2.4 Context-Awareness

The large number of sensors could be pervasive in a smart home within the IoT paradigm. It is difficulte to control each smart device for the user, a smart home requires certain specific kind of intelligence, the smart devices and sensors should be capable to adapt to the users changing habits. The smart devices generate raw data, in order to understand and benefit from those raw data, there is a need for algorithms that can integrate, model, reason. Context-aware computing is a system that can provide its users with relevant information about an environment, a person, device or an application, anywhere and anytime, using contextual information related to a users task (Abowd *et al.*, 1999). The first term 'context-aware' was used through (Schilit & Theimer, 1994). Context-aware computing is able to understand sensor data. (Perera *et al.*, 2014b) presented a survey study of the IoT paradigm and context-aware computing. Additionally, the study focused on the analysis of context-aware computing by reviewing 50 context-aware computing projects. Each project in this field, from 2001 to 2011, was critically evaluated and addressed proposed solutions to further research in this field. (Perera *et al.*, 2014a) conducted a survey that context-aware computing has played a critical role during the last decade of ubiquitous computing and also it will be expected to do a significant role in the IoT paradigm.

2.3 Smart Home

Home automation has existed for some time; now the technology is at a stage where individual households can make use of it. Home automation provides comfort, home energy management, security and can help the elderly and disabled to receive quality care (Bartram *et al.*, 2011). (Bartram *et al.*, 2011), proposed the design and implementation of an interactive system which provides citizen awareness of resource use. It provides simple and efficient management of a house system to enhance daily activities, but although this approach has great potential to help residents with sustainable living, there are many challenges as to how the technology is integrated with the home environment. A smart home includes sensors, actuators, middleware and a network.

2.3.1 Definition

A smart home connects different devices and pervasive computing systems in homes to deal with each other, which are automatically controlled remotely from the Internet. Many terms apply for smart homes, such as smart house, home automation, and

adaptive home(Alam *et al.*, 2012). There are many definitions of smart homes.

> "The smart home concept is the integration of different services within a home using a common communication system. It assures an economic, secure and comfortable operation of the home and includes a high degree of intelligent functionality and flexibility (Lutolf, 1992)."

This definition focuses on home automation and does not focus on the role of home intelligence. Another definition by (Berlo *et al.*, 1999) defines a smart home as

> "A home or working environment, which includes the technology to allow the devices and systems to be controlled automatically, may be termed a smart home. "

This definition covers the smart home domain that contains several devices and appliances that can be automatically controlled remotely. (Briere & Hurley, 2011) defined it as

> "a smart home as a harmonious home, a conglomeration of devices and capabilities based on home networking."

This definition is general and is not specific to the smart home idea. Intertek in 2003 published a more accurate definition to express a smart home:

> "A smart home is a dwelling incorporating a communications network that connects key electrical appliances and services and allows them to be remotely controlled, monitored, or accessed (Alam *et al.*, 2012)."

According to Intertek, there are three factors that a home requires to be smart:

- Internal network: A smart home contains various types of home appliances, sensors, and services that can be connected through the cable or wirelessly to the internal network.
- Intelligent control: A smart home gateway is integrated to connect different smart home devices to provide the homeowner with control of various applications.
- Home automation: Home automation enables various smart home appliances, sensors, and devices to link and work with services to provide intelligent home services outside the home.

An additional definition by (Satpathy, 2006) is more suitable for the smart home concept:

> "A home which is smart enough to assist the inhabitants to live independently and comfortably with the help of technology is termed as smart home. In a smart home all the mechanical and digital devices are interconnected to form a network, which can communicate with each other and with the user to create an interactive space. "

But does not focus on how it can be automatically controlled remotely. (Alam *et al.*, 2012) stated:

> "we can define the smart home as an application of ubiquitous computing that is able to provide user context-aware automated or assistive services in the form of ambient intelligence, remote home control or home automation."

This definition is inclusive for the smart home concept.

2.3.2 Classification

Smart Home Environments (SHEs) intersect with several research areas, such as computer networks, computer engineering, applied computing, and embedded systems. The focus of smart environments is identified with interactions between multiple agents, and the smart environment is able to present services that improve the performance of a smart home (Cook, 2009).

1. A home automation system includes many home appliances, electronic devices, and smart devices, such as washing machines, lighting systems, smoke detectors, etc. Whose functions may be divided into sensors, actuators, or both (Jih *et al.*, 2006).
2. A control system that combines a human user with software to control and manage the home that receives information from sensors and instructions sent to actuators to perform specific tasks in SHEs.
3. A home automation network has the capability for the smart home to connect and work among the home automation and control systems.

The SHEs contain two parts, hardware and software. (Badica *et al.*, 2013) illustrated a classification of SHE according to the following information discussed below.

2.3.3 Structure

The home automation system has a huge number of home appliances and electronic devices, which are selected based on certain applications. With the increase in new emerging technologies of home appliances and electronic devices, (Fortino *et al.*, 2012) presented a new smart object concept to explain sophisticated electronic devices and home appliances.
The smart object is divided into three parts: 1) the object in the real world represents the physical part, 2) the hardware infrastructure grants processing power to allow smart capabilities to the object, and 3) the software layer supplies smart capabilities. The home automation network contains a network component technology and communication protocols.
The network component can be of any type whether powerline, bus line, or wireless. Powerline communication is a communication technology that transmits information through current power cables, it reuses the in-home electrical network, it is low cost but less reliable. The bus line home networks are local area networks in which each device is connected to a main cable. It can be used via physical media, such as switches, hubs, and routers, to transfer electrical signals. Bus line home networks are simple to expand and are reliable. However, they suffer from some disadvantages for the home resident, require cabling systems, and are not easy and fast to set up and connect to the Internet. Wireless home networks are wireless networks that are not connected through any cables, such as infrared (IR) or radio frequency. The use of a wireless network avoids the costly cables and devices that provide convenience and mobility for home residents.
(Gomez & Paradells, 2010) presented several communication protocols that are used for home networks. There are three types: 1) proprietary protocols for private companies, 2) public protocols for the public, and 3) standard protocols through the standardisation organisation. There many home network protocols, such as IEEE 802.15.4/Zig-Bee, IEEE 802.11/Wi-Fi, and 6LoWPAN.
The control system is more complicated in the SHE. Smart homes require a system that is able to react to the inhabitants' behaviour for a particular task, such as controlling the temperature or reacting to a smoke alarm or leaving the oven on for too long, through combining reactive and proactive behaviours to provide new services to the smart home inhabitants to improve their quality of living. In the literature, there are many methods that are proposed to develop advanced control systems for the SHE, such as artificial intelligence and multi-agent systems.

- Wireless Sensor Networks
  Pervasive computing systems contributed by wireless sensor network (WSN) technologies provide the integration of technology into a smart home environment.

The WSN is a set of wireless sensor nodes that are small, consume very low power, and are used to monitor or change physical status, such as temperature, humidity, light, and sound (Pirbhulal *et al.*, 2017).

Every sensor node contains five units. A microcontroller component or unit is able to process observed data. The microcontroller plays an important role in sensor nodes to analyse and process observed data from its sensor node or data received from another sensor node. After the microcontroller has processed these data, it sends the information to a neighbouring or other sensor node. The memory unit is where the data and information are stored. Each sensor node has a tiny amount of memory, and there is a short period of time to store data before analysing and transferring the data to neighbouring or other sensor nodes. Sensor node units sense the environment. The sensor node is a tool that converts data from analogue to digital. This unit transfers data sent or received to a format that fits the nature of the data in the microcontroller unit. The sensor node receives a signal from the sensor and converts the signal from analogue to digital to send it to the microcontroller for more processing (Bhattacharyya *et al.*, 2010). Communication is one of the most important units or components of the sensor node that executes an exchange of the aggregated data from various sensor nodes. It is one of the most energy-consuming units due to transmitting and receiving data between various sensor nodes. There are many technologies to transmit data and information, such the standard IEEE 802.15.4/ZigBee, Bluetooth, and power supply units. Each sensor node requires a power supply that has a greater life. Normally, each sensor node is supplied with batteries, and they can be recharged. There are several factors that influence energy consumption, such as how long it takes for transmitting and receiving data and the environmental conditions in the surroundings, such as the temperature.

There are two categories of sensor nodes. A passive sensor is used to detect and reply to any change from environmental conditions, this kind of sensor is a microwave instrument. Passive sensors include two kinds of sensors, the omni-directional sensor is a sensor that detects any input from the physical environment without really manipulating it by active probing. This type of sensor requires energy just to amplify analogue signals, and each sensor is self-powered, which uses power from the environment, for example, light sensors, smoke detectors, microphones, pressure and parking sensors, etc. Passive narrow-beam sensors are another kind of passive sensor, which are able to use direction. This sensor detects a given direction of the physical environment, for example in a passive camera, which does not need to be supplied with energy. The passive narrow-beam sensors are used in smart home environments and are fit for internal applications, for

example monitoring, healthcare applications, and home automation. In a smart home, for example, when the inhabitant installs a pressure sensor on the couch and the inhabitant sits on the couch, the pressure sensor can detect this and emit a signal (Skubic *et al.*, 2009a), (Bhattacharyya *et al.*, 2010).
Active sensors are those sensors that are required to be supplied with energy, such as radar sensors and temperature sensors. Active sensors emit radiation to detect objects and places in the surrounding environment. Then, the reflected radiation from that object or place is detected by the sensor and is measured. The benefit of an active sensor is that it can measure the reflected radiation anytime. There is no specific time to measure it, and there is no need for a natural source of energy. In smart homes, one example of an active sensor is when the inhabitant installs temperature sensors to measure temperature in the home (Bhattacharyya *et al.*, 2010), (Yang *et al.*, 2010).

- Wearable Sensors
  Wearable sensor devices are used for assisting people to provide health care, for example, monitoring, managing home appliances, continuously monitoring medical status, emergency detection, etc. These devices can be integrated into a wearable wireless body area network, and the devices can be embedded or implanted in the human body. Different wearable or implanted items in the human body are able to collect a large amount of data to provide caregivers with useful and important information about the health status of their patients when this technology is integrated in the telemedicine system (Darwish & Hassanien, 2011), (Rashidi & Mihailidis, 2013) presented a study conducted on wearable sensors in smart home applications. There are several sensors, such as the accelerometer, gyroscope, and global positioning system (GPS), that are used for detecting inhabitant activities and mobility. Some studies have generated datasets from the smart home environment using a smartphone that collects the accelerometer and gyroscope data from participants (Casale *et al.*, 2011).

- Ambient Sensors
  Ambient sensors are several pervasive sensors throughout each room that are equipped with many sensors that include home appliances, electronics, light, and temperature. All sensors work with each other to capture the interactions of the residents (Pauwels *et al.*, 2007). Although ambient sensors are able to detect geospatial information on the inhabitant activities in smart home environments, ambient sensors are not associated with a specific location. Ambient spatial intelligence is used for tracking inhabitant motions and for finding the relation between spatio-temporal events in the environment (Duckham & Bennett, 2009).
  (Chen *et al.*, 2015) compared ambient sensing data and wearable sensing data.

The ambient sensors are stable and generate data that are more accurate than data found by wearable sensor devices on the human body. Many ambient sensors can be installed in rooms to identify activities with multiple residents, but wearable sensors employ a small number of sensors to collect data from the human body.

2.3.4 Architecture

The architecture is a system design that contains all of the hardware components within a computing system. With the recent rise of the IoT, ubiquitous and pervasive computing has become widespread, which requires an increase of computational power of devices, leading SHE architectures in a direction from being centralised to distributed.
(Ahsan & Bais, 2018) discussed the huge amount of data generated from sensors. There is a need to process the data, and there are two methods: a central computer or multiple distributed processors around sensors.

- Centralised Architecture:
  The control system is achieved through a computer system that is responsible for the data captured from smart homes by sensors, user interfaces, and applications of the control algorithms and the transmission instructions to the actuators. The central computer is called the home gateway. The central computer or the gateway allows the home inhabitant to control and manage the home. It is enabled for the smart home to connect and work with services outside the home via the Internet.
- Distributed Architecture:
  In a distributed SHE architecture, the computational resources of smart objects assist the distributed architecture to establish software into the nodes of the smart home network(Catarci *et al.*, 2008)

2.3.5 Projects and Applications

There are several research projects, applications, and uses for smart homes. (Alam *et al.*, 2012) offered the classification of smart home projects based on specific tasks: comfort, healthcare, and security. (Badica *et al.*, 2013) identified four major application areas of smart homes: (i) elderly/ageing/home care, (ii) energy efficiency, (iii) comfort/entertainment, and (iv) safety/security. Each area can be associated with other areas, such as security with home care.

- Comfort/Entertainment:
  This application is one of the main goals of smart home research to provide inhabitant comfort. There are two types to achieve comfort and entertainment for residents. The first is increasing the level of automation of routine activities through knowledge of the inhabitants' activities of daily living, habits, and behaviours. The second is remote control capabilities for home appliances and home tasks, such as turning up the air conditioner when the temperature is high in the home.

- Elderly/Ageing/Home Care:
  This application attempts to address the needs of elderly people due to the recent increase of the ageing population in the developed world. In a study conducted by the World Health Organization (WHO), the number of elderly people aged 60 or older will continue to grow and will reach 2 billion in 2050 (10 Facts on Ageing and the Life Course) [2]. Thus, some of them have lost mobility, sensorial, or cognitive skills that leave elderly people unable to live independently. Ubiquitous computing technologies can provide two services for security and independency. Although there are existing systems, the development of such systems in the home for safety is limited by the focus on only elder monitoring and emergency detection (Taleb *et al.*, 2009).
  (Alam *et al.*, 2012) illustrated a healthcare project in a smart home that provides healthcare support for elderly people and healthy people. There are two healthcare services. The first one can be applied on a website to produce health reports locally. The second one uses a remote healthcare service to provide and support elderly people in emergency situations.

  1. Local monitoring:
     Sometime elderly people require monitoring of certain activities of daily living to estimate health conditions. When something happens that is abnormal, the alarms send an alert to the medical office or homeowner. The research team at Tampere University of Technology, (Vainio *et al.*, 2008) proposed a proactive fuzzy home control system. An adaptive fuzzy logic algorithm is used to assess the test on the acquired results to help elderly people live autonomously at home. This system is able to recognise routines. The patterns gained from the daily routines will be beneficial for assisting elderly people. It provides inhabitants' information, such as sleeping disorders, medications, diabetes data, and blood pressure.

  2. Remote monitoring:
     Sometimes older adults need monitoring of certain daily activities to estimate

[2] http://www.who.int/features/factfiles/ageing/en/ (accessed 28November-2014)

a medical situation. Health care and safety are significant features for elderly people in the home. Those tasks include a fall detection mechanism, smoke alarm sensors, oxygen sensors, etc. Remote monitoring is used as a health-care service to provide their families and caregivers with useful and important information. The home observes the older adults using physiological sensors and communicates with their families and caregivers automatically in health emergency cases. Some smart home projects were discussed to provide remote patient-monitoring services. There is a need for real-time homeowner intervention from outside the home.

(Barnes *et al.*, 1998) proposed low-cost technological solutions to assist in the care of elder people through British Telecom and Anchor Trust in England, which monitored the activities of daily living of elderly people. It can be used to enhance the provision of care to elderly with reduced human intervention. Their technique discovers the occupants' movement using IR sensors and magnetic communicators at the entrance of the household doors. There are many sensors used in the home, such as temperature sensors and alarm sensors, which detect anomalies and contact the control centre or homeowners.

3. Safety/Security:
   Safety is an important application of a smart home. Safety mechanisms identify abnormal behaviour by monitoring the inhabitants' daily activities in the SHE, for example, fire, falls of older adults, and slow movements. Techniques can be used to find correlations between spatial and temporal aspects of the dataset to detect abnormal situations (Ni *et al.*, 2015). Security detects malicious behaviours in the SHE, such as unauthorised access for malicious purposes, user impersonation, thieves, and privacy violation. Smart homes are more susceptible to security and privacy threats. There are many anomaly detection techniques that can be used to identify an abnormal behaviour. It is one of the main issues in this domain. Due to the nature of the data, a SHE contains huge volumes of data that need to be inspected. However, there are tools and systems that can be used to increase the security in a smart home, such as video surveillance, remote monitoring, alerting, and alarms.

- Energy Efficiency:
  Energy saving is a significant topic because of frequent climate change. There is a universal direction and request for energy saving to increase the effectiveness of energy consumption. Smart home technologies play an important role in increasing energy efficiencies (Bhati *et al.*, 2017). Recently, the technological advance demands usage of more energy. However, there is limited energy from the lack of

resources. In Europe, 40% to 45% of power consumption occurs in buildings and from inhabitants' activities (SBCI, 2007). Consequently, household energy savings in smart homes can be understood as using technology in energy savings.

Home automation provides a means for energy management and supports feedback on consumption at device level. Recently, (Harle & Hopper, 2008) proposed the use of sensing networks to analyse and respond to user behaviour in smart environments to improve energy use management.
With the focus on energy consumption, (Weiss *et al.*, 2009) proposed an interactive system that provides instantaneous feedback on energy usage in the household. They have uploaded the application to the internet in order to provide power management that enables the user to monitor consumption on a smartphone.
(Pipattanasomporn *et al.*, 2012) focus on intelligent home energy management in a smart home environment, using a home energy management (HEM) algorithm in order to demand responsive applications. Their results showed that the HEM algorithm enables proactivity and facilitates efficient control, managing appliance operation to keep the sum household consumption under a specified demand limit.

In addition, a new smarter network is called the smart grid. It increases the normal electricity delivery system by using more sophisticated communication and information technologies to reduce the power consumption. Additionally, it attempts to exploit the potential of renewable energy sources (Saponara & Bacchillone, 2012). (Badica *et al.*, 2013) specified two main application areas regarding to energy consumption:

1. Energy saving using smart technologies in the smart home is aimed to decrease energy consumption and enables the occupant or control system to control the power reduction at home by turning off all appliances that are not in use, by turning them on when the inhabitants need them, or by setting the devices at a low-power status. After determining the settings for energy at home based on the inhabitants' activities of daily living, the settings can be controlled locally or remotely. Smart homes allow homeowners to control the devices, appliances, and systems automatically (Torunski *et al.*, 2012).
2. The smart grid is the electrical network in which communication technologies are used, such as smart meters that works with 4G networks to collect information from sensors, consumption points, and power generators. This method is integrated into the network management. It is automated to improve the efficiency and reliability of the generation and transmission of electricity to the electricity consumers (Speer *et al.*, 2015). The smart grid

exploits technologies that improve network fault detection and network self-repair without the intervention of technicians. It provides the electricity price to the consumer to reduce power consumption at peak times.

(Jahn *et al.*, 2010) presented a system that integrates into the SHE energy efficiency features using the Hydra 1 middleware framework. They used this framework to settle the heterogeneous issue, which offers the challenge of executing interoperation among devices in a SHE. It uses a wireless network to take data from home appliances and sensors and combines the wireless power sockets to reach a target power consumption. It can use the data for monitoring, controlling, and analysing the consumed energy in each device. It automatically provides the homeowner with all details, such as the electricity price, how much energy is used, and the meter readings.

2.3.6 ADLs Monitoring

One of the projects that used supervised techniques is MavHome which aims at creating a smart home environment that acts as an intelligent agent by reading the sensors' data and intelligently manipulating the environment using device controllers. The project proposes a scalable architecture to achieve goals. Each agent in MavHome architecture is composed of four layers, which are the decision layer, the information layer, the communication layer, and the physical layer. The relevant layer to the research at hand is the decision layer, which is responsible for deciding the agent's action based on the gathered information. The project uses a Smart Home Inhabitant Prediction (SHIP) algorithm which works by searching and matching recent sequence of events with previously captured sequences. They have evaluated their proposed algorithm, on a real dataset and it scored a classification accuracy of 53.4% and 94.4% on a synthetic dataset, they have been used their own datasets, which is not available in public domain. The simplicity of the SHIP algorithm is one of its strengths. However, it has a limitation of not being able to operate in an online fashion. The whole historical activities must be stored and processed offline. To overcome this limitation, the project developed Active LeZi (ALZ), which is a sequential prediction algorithm. They tested this algorithm performance on a synthetic dataset and it scored 87% accuracy. Moreover, the project applied a Task-based Markov Model (TMM) and it scored 74% accuracy on a 30-day synthetic dataset (Cook *et al.*, 2003) (Lesser *et al.*, 1999b) (Das *et al.*, 2002).
(Hussein *et al.*, 2014) developed a theoretical design for disabled people in a SHE. The simulation was conducted to capture activities of daily living data from their SHE for elderly or disabled people in real life, their dataset is not available in public domain. They proposed the use of a neural network and a Recurrent Neural Network (RNN) because

of the suitably of RNN to handle sequential data. Neural network testing result accuracy was 95% in the fire alarm prediction, while the user behaviour prediction accuracy approached 80%.

(Assim *et al.*, 2006) presented a prototype of a system in a SHE, which can make decisions without human intervention, adapt to daily living activities, and provide automation for smart home appliances to achieve the inhabitants' quality life. It can cooperate with multi-agent systems to share knowledge. Multi-agent systems have more benefits when there is more than one agent. Each agent performs various tasks and can interact, share knowledge, which has a worthy effect on the system performance (Alam *et al.*, 2012).

(Lesser *et al.*, 1999a) designed and implemented a smart home simulation with multi-agent systems to control an intelligent environment, where each agent is distributed and performs a specific task. The agents interact and share resources and knowledge with each other and use various coordination protocols to distribute tasks among them. It can adapt to changes in the environmental input.

(Mozer *et al.*, 2005) developed an adaptive house to monitor elderly people in a SHE. This adaptive home aims to adapt proactively to changes to assist the inhabitants by learning and observing their activities of daily living to meet their needs. They used a neural network algorithm. They described the adaptive home that facilitates controlling the temperature of each room and the lighting separately to provide a convenient environment and save energy for inhabitants, but there were limited applied artificial intelligence features. Due to the limited use of neural networks, although Artificial Neural Networks (ANNs) have more potential. (Begg & Hassan, 2006) indicated that ANNs have effective capabilities to work particularly with the automated monitoring and manage the devices and home appliances in a smart home.

(Chan *et al.*, 1995) developed a smart home automation system for observing elderly and disabled people. The ANN was used, and they used different sensors that were installed in 12 rooms. The dataset represents real activities captured from the residents to evaluate the ANN algorithm to classify activities of daily living in smart homes and to detect anomalous behaviours. Moreover, the convergence rate for the ANN was reported at 90%, their dataset does not available for researchers.

(Mozer, 1998) developed a smart home automation system to manage essential convenience systems to save energy, such as lighting, air heating, water heater, etc. This system is programmed to learn and monitor inhabitants' activities of daily living in a SHE and to predict their activities. The ANNs have been used to predict these actions, but there is a lack of interaction among the users and the proposed system; all these systems are becoming automatic.

(Cavone *et al.*, 2011) proposed an agent approach to manage normal behaviour in SHEs.

Their system used the butler agent that can be employed to learn and infer from inhabitants' activities. It can automatically adapt proactively to changes of the user. The system provides interaction among users and the system by presenting some services to the user that can be accepted, cancelled, or modified. The butler benefits from the user feedback to improve future conduct. This system can be divided into four categories:

1. The sensor agents present information about the user context and sensors, such as temperature, lighting, air heating, water heater, etc.
2. The butler agent manages the smart home. Its learning, reasoning, and inferencing model works to achieve the desires of the user, and it chooses the workflow related to certain targets and needs.
3. Effector and interactor agents for controlling devices and actuators can be controlled automatically or controlled between the user and the system.
4. The housekeeper agent works as a mediator and identifies all active agents in the home, and it knows what each agent is able to achieve.

In the real era of smart homes, there are many technologies and applications that allows efficient communications between home appliances and inhabitants and enables automation, monitoring, and remote control capabilities for home appliances and home tasks (Lobaccaro *et al.*, 2016).

The Episode Discovery (ED) prediction algorithm was used to identify repetitive events to automate the next series of events (Dixit & Naik, 2014). (Gopalratnam & Cook, 2004) research focuses on the role that predictions play in the field of artificial intelligence and machine learning for creating smart systems that are able to make decisions more reliably. The smart home environment requires the ability to predict a sequence of events. They used Active LeZi (ALZ), which is a sequential prediction algorithm. They tested this algorithm performance on their own synthetic datasets, which are representative of interactions among inhabitants and the smart home. Machine Intelligence Algorithms include a variety of techniques, technologies and methods. This research aims to propose a machine learning algorithm that can learn the inhabitants' daily living activities that can predict such activities using supervised learning techniques applicable in smart home environments. There are some research studies and research surveys that have covered most aspects of a smart home. e.g. (Amiribesheli *et al.*, 2015) presented their survey of many smart home research studies and explained some algorithms.

2.4 Machine Intelligence Algorithms for Smart Homes

Machine learning has been widely applied to develop probabilistic and statistical methods and sequence-learning algorithms to predict activities of daily living (ADLs) of inhabitants. Machine learning techniques can be divided into three categories based on the availability of labelled datasets, as such:

- **Supervised** learning is the process of an algorithm learning from a training dataset with labels, this technique is used when there is full availability of the ground truth labels. This is because, in smart homes, most homes and home appliances have extremely variable layouts, and residents conduct their activities in various ways. The same activity might use completely different sensor activation depending on the inhabitant's changing habits; therefore, to deal with variability, supervised learning is used, which is mostly the case in synthetic datasets that are generated using simulation tools.
- **Semi-supervised** techniques are used when parts of the ground truth labels are available. This is usually the case in real-world datasets because the inhabitants are asked to record their activities manually. This approach is prone to human errors.
- **Unsupervised** techniques are used when there are no ground truth labels available for the ADLs. In this situation, clustering techniques are used to group similar ADLs into clusters. However, clustering techniques alone do not classify and predict ADLs. They are usually used with other techniques to facilitate certain aspects of the learning model, such as performing a pre-processing step of the data (Tapia *et al.*, 2004) (Bourobou & Yoo, 2015).

This thesis used supervised learning techniques for activity classification in a smart home. These techniques require the full availability of the labels to train from because most homes have changing layouts, and inhabitants perform their activities in various ways. The same activity might use a completely different sensor activation depending on the inhabitants changing habits. To deal with variability, there is a need to use supervised learning techniques in smart home environments. (Tapia *et al.*, 2004) illustrated using supervised learning when there are labels available for the ADLs, which provides a promising approach to recognise the activity in a smart home.

Machine learning is a technology that enables algorithms to learn from experience in an existing dataset. When there are massive amounts of data, machine learning learns

from the training data to recognise complex patterns and make prediction decisions. Machine-learning techniques have three main approaches that will be discussed in the following points.

- A classification problem is a supervised learning approach in which the models learn from a training dataset with labels and then use the model to classify new labels for the data. The model can predict and label an unseen example with a class. The output labels could use a binary classification, such as a person is male or female or mail is spam or not spam, and it could use a multiclass classification, such as activity labels in a smart home or in speech recognition. The input data are classified into discrete linear or nonlinear categories:

  1. **Linear classification or logistic regression**: Individual features are fed to the model to obtain a binary output, such as 0,1 or yes/no. These types of problems are called binary classification problems.
  2. **Nonlinear classification**: When features are fed to the model to predict output that uses more than two label classes, These types of problems are called multi-classification problems (Gunn *et al.*, 1998).

  In classification tasks, the classifier should learn from input variables in the dataset to predict discrete values for output variables. The classifier predicts the label or class with the highest correlating probability for new observations. For predicting the weather, it could record the weather forecast of the city for the whole day regarding whether is it going to rain today or not. Another application in a classification problem is when the mail service uses the classification technique to classify the mail as spam or non-spam. The classifier trains itself on a dataset to analyse user behaviours for a long period of time. Relying on this knowledge, the classifier can predict whether new mail belongs to the non-spam class (the inbox folder) or spam class. As this research mentioned, the model is able to use the classification to predict more than two classes, for example in a smart home environment that includes multiclass classifications, such as sleep, eat, personal, work, and leisure. There are many types of classification algorithms in machine learning, such as support vector machines, decision trees, neural networks, etc (Alpaydin, 2009).

- A regression problem is a supervised learning approach that tries to predict a continuous value. The regression is used to estimate the relationship between variables. The regression is used to predict the numeric data, such as temperature or price rather than discrete categories and labels, for example predicting a house price from its space or the salary for a job based on its advantages. Thus, the relationship between the variables is estimated by a linear function. The regression is

similar to the classification problem, attempting to evaluate a function that maps the input to the output depending on early observations. However, the regression problem attempts to evaluate a real number or value instead of classes or labels. There are several kinds of regression algorithms. They can be used based on the type of problem. Linear regression is used when research has one feature; for example, it predicts the price of a house knowing its square feet. However, researchs sometimes require more details about the house, such as the number of bedrooms, bathrooms, city, etc. To predict the value, there is a need to add two or more other variables, so multiple regression is used. There are some regression algorithms that have the ability to deal with high-dimensional data. Multiple linear regression is used when research has several different variables, such as x1, x2, ..., xn. In a polynomial regression, it uses the same variable x1 but with different powers, so instead of x2 it has $x1^2$. It uses a variable with different powers. There are several observations and then a line that fits the data. Sometimes simple linear regression is not suitable to rectify that using a polynomial regression. There are many types of regression algorithms in machine learning, such as linear regression algorithms, multiple regression algorithms, polynomial regression algorithms, and least-angle regression algorithms (Gunn *et al.*, 1998), (Pedregosa *et al.*, 2011).

- A clustering problem is an unsupervised learning approach that that uses clustering to group similar datapoints into sets. Clustering aims to assign similar datapoints in the same cluster and different datapoints into different clusters. The clustering technique does not have labels available that can be used for training phases. There is a concept of distance between the datapoints to group the datapoints together into many clusters that are similar. Clustering works with low and high-dimensional space, and it can determine the similarity using a distance measure (e.g. Euclidean, Manhattan, or Minkowski) (Marsland, 2011), (Pedregosa *et al.*, 2011).

  However, the clusters suffer from disadvantages. In cluster problems, the ability to separate classes from each other exists, but in high-dimensional space, they will sometimes overlap with each other, datapoints mix with each other, and it is difficult to locate obvious boundaries among the clusters.

  The k-means algorithm divides a set of datapoints into separate clusters. The algorithm tries to calculate the average of the datapoints in the cluster by choosing an initial centroid location in the middle of each cluster. It allocates every datapoint to the cluster based on the nearest centroid by distance, such as Euclidean distance. It iterates the same steps until the k-Means algorithm does not make a new cluster, which is called a stable cluster (Marsland, 2011).

The spectral clustering algorithm is an unsupervised learning algorithm that attempts to decrease the dimensions of the datasets to reach affinity among datapoints, so the algorithm can represent the datapoints. This technology is easy carry out. It uses standard linear algebra, so it offers good solutions (Von Luxburg, 2007). The cluster technique is used in many fields in machine learning, such as pattern recognition, image analysis, and information retrieval. There are many types of clustering algorithms in machine learning, such as the mean-shift clustering algorithm, k-means algorithm, and spectral clustering algorithm.

In the real era of smart homes, there are many technologies and applications that allows efficient communications between home appliances and inhabitants and enables automation, monitoring, and remote control capabilities for home appliances and home tasks (Lobaccaro *et al.*, 2016). A project focusing on a smart home in an intelligent environment is MavHome. This project describes an architectural style and the role of prediction algorithms within the architectural design. This type of design requires prediction algorithms to foresee the inhabitant's next action to automate the recurrence tasks for the inhabitant (Cook *et al.*, 2003). This model of a smart home will be able to make decisions and increase automation to achieve inhabitant‘s comfort and save energy. Consequently, the focus is on prediction algorithms that can recognise the next event. The Episode Discovery (ED) prediction algorithm was used to identify repetitive events to automate the next series of events (Dixit & Naik, 2014). (Gopalratnam & Cook, 2004) research focuses on the role that predictions play in the field of artificial intelligence and machine learning for creating smart systems that are able to make decisions more reliably. The smart home environment requires the ability to predict a sequence of events. They used Active LeZi (ALZ), which is a sequential prediction algorithm. They tested this algorithm performance on synthetic datasets, which are representative of interactions among inhabitants and the smart home. Machine Intelligence Algorithms include a variety of techniques, technologies and methods. This research aims to propose a machine learning algorithm that can learn the inhabitants' daily living activities that can predict such activities using supervised learning techniques applicable in smart home environments. There are some research studies and research surveys that have covered most aspects of a smart home. e.g. (Amiribesheli *et al.*, 2015) presented their survey of many smart home research studies and explained some algorithms.

2.4.1 Support Vector Machines (SVMs):

Support Vector Machines (SVMs) are supervised learning models used for classification and regression analysis (Cortes & Vapnik, 1995). Also, SVMs have many benefits, high

dimensional feature space. SVMs are effective when the number of samples is less than the number of the dimensions in the dataset. Moreover, SVMs can be efficient on memory usage. SVMs can be used with different kernel functions which will allow the model to learn complex decision function. On the other side, SVMs have some disadvantages such as over-fitting, which can occur when the number of features is greater than the number of samples (Pedregosa *et al.*, 2011).
SVMs have been used in the literature for the classification of ADLs in the health domain. Health Smart Home is one of these efforts which includes real data collected from various sensors, to evaluate the SVM algorithm. (Fleury *et al.*, 2010) installed many sensors and microphones in the environment and obtained a classification accuracy of 75% and 86% using a polynomial kernel and a Gaussian kernel respectively .

2.4.2 Hidden Markov Model (HMM):

Hidden Markov Model (HMM) is an unsupervised generative probabilistic model. The HMM deals with hidden states, which means the state is not observed directly. The transition from one hidden state to another can be modeled as a Markov process. The HMM is suitable for sequential datasets. States have a probability distribution on the likely output symbol (Baum & Petrie, 1966).
(Alemdar *et al.*, 2013a) applied a HMM to classify ADLs of smart home multi-inhabitants. They proposed the use of HMM because it appropriately handles sequential data. The hidden states were modeled to be the activities' labels and the observations are the sensors' readings. To evaluate the accuracy of the model, They had been developing the ARAS dataset. The dataset represents real-world activities captured from multi-residents in two real houses. HMM average accuracy was 61.5% in house A, while house B the average accuracy approached 76.2%.
(Babakura *et al.*, 2014) applied a HMM based decision model in smart homes in order to solve heterogeneous problem that presents the challenge of executing interoperation among devices in smart home environment. HMM is able to learn such subsystems. They used the dataset from events that occurred in a smart home building that contains more than 6000 events, obtaining a classification rate accuracy of 95.7%.

2.4.3 Decision Trees (DT):

Decision Trees (DT) are supervised non-parametric learning models used for classification and regression. Non-parametric learning models do not assume that a probability distribution generated the data. A DT model learns simple condition rules inferred from the labelled data. Thus, a DT model is easy to interpret and understand. Moreover, the

model usually offers good performance and its time complexity is low. Some machine learning algorithms cannot work with certain data types. However, Decision Trees are able to work with categorical and numerical datasets alike. For multi-label classification problems, DT can work and offer good solutions. Very little data pre-processing is required when creating these models. However, DT suffer from some disadvantages. DT cannot work with missing values without preparing the dataset. They are prone to over-fitting and can produce complex models that are sensitive to small changes in the dataset, which do not generalise well and may produce unstable models. There are some datasets that can be hard for the DT to learn. Moreover, it is not guaranteed that the learned DT is the optimal tree (Pedregosa *et al.*, 2011).
DT can be used to classify ADLs of smart home inhabitants. E-ID5R is an extension of the DT algorithm to allow it to work with multi-label classification problems (Prossegger & Bouchachia, 2014). The accuracy of E-ID5R was evaluated using the same ARAS dataset (Alemdar *et al.*, 2013a). E-ID5R classification accuracy approached 40% on house A and 82% on house B.

2.4.4 Stochastic Gradient Descent (SGD):

Stochastic Gradient Descent (SGD) is an iterative algorithm used to find the minimum and maximum value of a function. Usually, it is used with convex loss function to find the minimum error. It can be used with linear classifiers, such as SVMs, for classification and multi-label classification problems. It is able to work and scale with large datasets. However, it needs several hyper-parameters to be set, such as the learning rate and the number of iterations (Pedregosa *et al.*, 2011).
The logistic regression with SGD algorithm were applied in order to develop a scalable diagnosis model for health care applications. To assess the proposed algorithm, they used the Cleveland Heart Disease Database (CHDD) which collected data from wearable body sensors used to measure the blood pressure and heart disease rate. Logistic regression with SGD algorithm enabled the model to predict and classify the heart disease status. The accuracy of training and validation on the data sample was 81.99% and 81.52% respectively (Manogaran & Lopez, 2017).

2.4.5 AdaBoost:

AdaBoost is a supervised learning algorithm used for classification and regression. The algorithm uses a group of weak learners or weak prediction models. The final prediction is the result of all the predictions from the weak learners combined. Thus, the model can be thought of as a majority voting system. AdaBoost can be used for multi-label

classification problems as well (Freund & Schapire, 1997; Pedregosa *et al.*, 2011).
(Logan & Healey, 2006) proposed the use of AdaBoost algorithm to deal with the classification of eating and meal preparation in the smart home. In order to reduce the number of sensors and focus on using the main sensors required for this task, they used the dataset from the MIT PlaceLab project. The authors used only 8 sensors from over 300 sensors, obtaining a rate accuracy of 82%.

2.4.6 Hierarchal Temporal Memory (HTM):

The HTM theory attempts to model the architecture and structure of the neocortex, the front part of a human brain. The focus of the theory is on the neocortex because it is envisaged to be where the human intelligence resides. The CLA is a machine learning algorithm that is based on HTM theory, which aims to explain the structural and algorithmic characteristics of the neocortex (Hawkins & Ahmad, 2016).
The HTM theory and its algorithmic implementation, the CLA, have been applied in many domains. Such as vision (Škoviera & Bajla, 2013; Arel *et al.*, 2010), natural language processing (NLP) (Webber, 2015), and anomaly detection in smart homes (Otahal & Stepankova, 2014). (Otahal & Stepankova, 2014) presented a study, which used HTM and its CLA to classify "healthy" and "sick" patients using a dataset that contains 70 patients. The dataset captures Electrocardiography signals (ECG). The CLA performance was slightly better than the multi-layer neural network, they illustrated benefits to use CLA that no preprocessing and It takes no more time to training. (Lee & Rajabi, 2014) conducted a study of the Numenta Platform for Intelligent Computing (NuPIC) that used HTM and its CLA for prediction and anomaly detection. They used the real-world DASHlink aviation dataset. In the evaluation, they compared the CLA against Scikit-learns linear regression, obtaining a prediction with more accuracy than CLA, and anomaly detection, obtaining a low accuracy rate. They recommended attempting to understand HTM and its CLA to modify its performance, although HTM and its CLA are still being developed (Lee and Rajabi, 2014).

(Zhang *et al.*, 2017) used the HTM system to propose an understanding action integration framework for skill learning. The model receives the input data from an RGB-D camera. The encoding region converts the input data to SDRs, and the HTM system learns the SDRs and predicts the future sequences. The accuracy of the proposed framework was evaluated using the shaking hands skill on a humanoid NAO robot.

(Škoviera & Bajla, 2013) applied an HTM to classify images that use colour features instead of images and features at the grey level. The HTM performance was good. (Mattsson, 2011) proposed the use of the HTM system to recognise images. Some items,

such as vegetables and fruit in a store, do not have barcodes for self-scanning. The main image converts to a binary array that is fed into the HTM. The HTM system is able to capture the colour, size, spatial space, and rotations of two kinds of fruit, obtaining a prediction rate accuracy of 97.5%.

2.4.7 Multi-layer Perceptron (MLP):

A Multi-layer Perceptron (MLP) is a feed-forward artificial neural network model, it maps values of input data onto a value of suitable output. It contains multiple hidden layers which are between the input and output layers. Each node is a neuron and every layer is fully connected to the next layer via weights. For each neuron, a weighted sum is calculated from the previous layer and then the result is passed to an activation function. After applying the activation function, the result is passed to the next layer. There are several types of activation functions, such as the sigmoid function, the hyperbolic tangent function and the softmax function. The MLP model uses back propagation for training the network in order to reduce the error (Rosenblatt, 1961). The structured perceptron is an extension of the standard perceptron that can predict structured data and usually it is used with an inference algorithm, such as the Viterbi algorithm (Collins, 2002; Zhu *et al.*, 2008).
The Back Propagation Neural Network (BPNN) has been applied to classify ADLs in a smart home. To assess the proposed algorithm, they used The Centre for Advanced Studies in Adaptive Systems (CASAS) that is a project for creating real smart homes for the researchers in this field. They demonstrated that the size of the neurons play important role to reduce the error rate (Fang & He, 2012).

2.4.8 Long Short Term Memory (LSTM):

Long Short-Term Memory (LSTM) is a Recurrent Neural Network (RNN). The LSTM model is good for classifying and predicting sequences, such as recognition of speech and handwriting. In regular RNNs, it is hard to train the model when the dependency of prediction has been seen a long time ago. This problem is known as the "long-term dependency problem" (Bengio *et al.*, 1994). LSTM is an extension of RNNs to overcome this problem (Hochreiter & Schmidhuber, 1997).
Deep convolutional and LSTM units framework was proposed in the domain of Human Activity Recognition (HAR) (Ordóñez & Roggen, 2016). It was used deep convolutional to extract special features from sensors data and LSTM to model temporal dynamics. The proposed framework was validated on two datasets, Opportunity dataset and Skoda

dataset (Roggen *et al.*, 2010a; Zappi *et al.*, 2008). The framework obtained 96% F1 score on the Skoda dataset and 93% F1 score on the Opportunity dataset.

2.4.9 Convolutional Neural Network (CNN):

Convolutional Neural Network (CNN) work like other neural networks but have a different construction. They are usually used to recognize visual patterns through images and videos. CNN can work with image data without requiring pre-processing for the data (LeCun *et al.*, 1998). CNNs were applied in different domains such as natural language processing (NLP) and recommender systems.
(Zeng *et al.*, 2014) proposed the use of CNNs algorithm to recognize the inhabitants' ADLs. The CNNs were able to capture local dependency of the activities and showed good scale invariance. Three datasets (Skoda, Opportunity, Actitracker) were used validate the proposed technique. The technique accuracy is 88%, 77%, and 97% on Skoda, Opportunity, and Actitracker respectively.

2.5 Intelligence

The term artificial intelligence (AI) has now become widely used. Moreover, AI is defined as the intelligence that machines and programs show that mimic the human brain and its working methods, such as the ability to learn, infer, and respond to unprogrammed events in the devices. This means how to make computers, software, and machines exhibit intelligent behaviour. In 1956, John McCarthy presented his definition of AI as "the science and engineering of making intelligent machines" (McCarthy *et al.*, 2006). At a conference on the campus of Dartmouth College in 1956, the modern field of AI research was founded, and the presenters and attendees Allen Newell (CMU), Herbert Simon (CMU), John McCarthy (MIT), Marvin Minsky (MIT), and Arthur Samuel (IBM) became the leaders of AI research for many decades (Russell & Norvig, 2016). The Dartmouth proposal in 1956 at the Dartmouth Conference was that each aspect of learning or any other feature of human intelligence can be simulated by a machine (McCarthy *et al.*, 2006). If the machine works intelligently like a human, its intelligence is similar to human intelligence. Alan Turings theory offered that, ultimately, we can only prove the intelligence of a machine based on its performance. This theory forms the basis for the Turing test (Turing, 2009). Hawkins developed a theory that explains how the human brain works in order to combine neuroscience and computing that enables a modern understanding of intelligence itself, which assists in creating intelligent machines (Hawkins & Blakeslee, 2007).

2.5.1 What Is the Type of Intelligence That a Smart Home Needs (Requires)?

Currently, smart sensor networks have been extended from their past capabilities. Consequently, the Internet of Things has been further developed. Smart sensors will interact without human intervention, via machine-to-machine communication. Each sensor streams huge volumes of data, and this data is processed in order to provide homeowners with useful information. This would allow them to be alerted if an event takes place and to make decisions as required (Chen, 2012). (Evans, 2011) presented a study about the relations between the number of devices connected to the Internet and the world population. In 2003, the world population was 6.3 billion people and devices connected with the Internet numbered more than 400 million, but in 2010, the world population was more than 6.8 billion and devices connected with the Internet numbered more than 12 billion. Obviously, there is an increased number of sensors that will generate a huge volume of data. In order to work with streaming data, which has changed over time, there are requirements to identify types of intelligence that are needed by smart homes.

2.5.2 Machine Learning Algorithms Requirements

The requirements for machine learning algorithms to classify and predict events in smart homes can be summarised as:

- to integrate and learn patterns from different sensors and to predict future events,
- to classify and predict events in a smart home domain,
- to be able to learn from historical events and attempting to predict future events is an essential requirement for smart homes,
- to find correlation between spatial and temporal aspects of the data set.
- to automatically deal and adapt with changes in the sensory input, without parameter tuning,
- to be more robust to noisy data which is common in smart home settings.

2.5.3 Research the Gap

Most of the previous work has targeted specialised environments for smart home, they have basic functionalities, nowadays smart homes require intelligence as mentioned in

next section. Important research in the area of the IoT has been carried out, particularly with regard to the influence and spread of networked computing on automation and the enabling of supportive and acclimated services in smart homes. The majority of previous work has targeted specialised environments for in-home care, such as the context-aware home (Kientz *et al.*, 2008).

There are many research efforts concerning a smart home environment, with some research focused on using a machine intelligence technique in a domestic environment. However, existing machine intelligence techniques do not fulfil effectively, all smart home requirements. As (Minor *et al.*, 2015) concluded, some algorithms in the smart home domain are subject to error when predicting the inhabitants' behaviour, when multiple inhabitants are living in the same home, multi-class classification is needed which is challenging and difficult task. Due to the nature of the data a smart home environment, there are multi-dimensional features available from the difference sensors. The data readings from sensors are always noisy and subject to many uncertain variables such as missing data and faulty sensors. Finding relationships between spatial and temporal aspects of the sensors readings to achieve high accuracy classification is needed (Minor *et al.*, 2015; Fatima *et al.*, 2012). The evaluation of the machine learning algorithms is limited by the lack of standard real datasets from smart homes that are publicly available. Due to the high cost of building real smart home datasets, there is a need for powerful simulation tools that can represent the activities daily living of inhabitant's (Cook *et al.*, 2009; Bouchard *et al.*, 2010; Synnott *et al.*, 2015). These simulation tools offer flexibility, scalability and accessibility for the researcher (Alshammari *et al.*, 2017; Bouchard *et al.*, 2010). This project proposes the use of HTM theory and its CLA implementation, as it is capable of mimicking the neocortex and hence can learn the inhabitant's activities, and it can predict future events and find meaningful spatio-temporal relations of high-dimensional data. The hypothesis is that HTM and its CLA has the capability to integrate and learn patterns from streaming dataset acquired from different sensors to predict future trends. This is what is required for a machine intelligence technique to integrate information from the smart home environment.

2.6 Why is CLA used for smart home

HTM theory is implemented within NuPIC, which is an open source project based on HTM. The HTM code in NuPIC can be used to analyses streaming data. (Cui *et al.*, 2016) conducted a comparison of the HTM model with other state-of-the-art algorithms in (neural networks). The model illustrates features that are important for sequence

learning, including online learning, and is able to deal with multiple predictions and robustness to sensor noise, as well as fault tolerance. HTM sequence memory is not specifically about how the brain works and solves problems, but also is usable to classification, prediction and anomaly detection.

2.6.1 Online Learning

HTM is an online learning system that works by learning and predicting in one step or multiple time steps in the future, it works with the streaming of data that changes over time, it does not need conventional training or testing data sets that leads to improve the rate accuracy over time, obtaining high results. The smart home has many sensors, and each sensor generates a huge volume of data. Smart home requires machine intelligence algorithms that can deal with the streaming of data which has changed over time. NuPIC is a memory system that is able to learn and predict, there is no requirement to save all of the streamed data.

2.6.2 Noise Tolerance

Due to the nature of the data in smart home, it is noisy. HTM theory has Sparse Distributed Representations (SDRs) to encode the data. SDRs are one of principal components of the HTM theory. They are biologically inspired by the neurons activities in the neocortex. SDRs might lose a lot of columns and cells, however it is able to produce the same predictions. One of their most important properties is their capability to resist noisy data, even if the data is noisy or warped, it grants the exact same output to the temporal memory.

2.6.3 Robustness and Generalisation

Most machine-learning algorithms require optimising a set of hyperparameters tuning for each task, so they should automatically handle problems without parameter tuning (Held *et al.*, 1996; Sharma *et al.*, 2000). HTM does not require any parameter tuning because HTM uses the same parameters, which were selected according to known properties of real cortical neurons, and the same cortical region is to be used for multi tasks (Hawkins & Ahmad, 2016). NuPIC is able to work with many different scenarios, it is also able to specify settings from one house and generalise them to other houses.

2.6.4 Conclusion

This chapter reviewed the IoT literature, smart home applications and machine intelligence algorithms typically used in the smart home arena. A comprehensive review of the state-of-the-art machine learning algorithms was executed and their advantages and disadvantages discussed. To classify and predict ADL of the inhabitant's in smart home environment, there is a need for machine learning algorithms that possess a level of intelligence that can automatically adapt itself to changes in the user ADLs. The machine learning technique should be able to cope with both streaming and noisy data. In order to test and evaluate the proposed machine learning techniques, there is a need to have a dataset which represents the ADL in a smart home (For more details on datasets see Chapter 4). The HTM and its CLA have the capability to integrate and learn patterns from a streaming dataset and find meaningful spatio-temporal relations in high-dimensional data. As the topic and associated algorithms has been discussed, it is proposed that HTM will become the preferred algorithm, the chapter 3 investigates HTM in details.

Chapter 3

Hierarchical Temporal Memory

3.1 Introduction

Humans have five senses: touch, smell, vision, taste and hearing, through which they are able to speak a language, understand spoken languages, recognise visual patterns and so on. However, when comparing a human with a computer, they are very different, and the computer is unable to perform exactly like a human. Although modern computers have a number of existing technologies such as Neural Networks (NN), it is a computing system; its contents are many processing elements (neurons) that are connected with each other. They process information via their changing state when receiving external inputs (Robert, 1989), Artificial Intelligence (AI) combines the science and engineering in order to make smart machines. Where there are many tasks using computers to understand human intelligence (McCarthy, 2007), Support Vector Machines (SVM) are part of machine learning in order to group classification problem that related learning algorithms. It analyses data used for classification which means supervised, unsupervised machine learning, and regression analysis (Cortes & Vapnik, 1995), Artificial Neural Networks (ANN) are formed for information processing, mathematical or computational model, it is inspired by biological neural networks such as the brain. The ANN is able to learn by example, such as pattern recognition in large of data or data classification, use artificial neurons to mimic the human brain (Siganos & Stergiou, 1996), and Deep Learning is a new branch or part of Machine Learning research, where the aim is to combine Machine Learning with Artificial Intelligence. It is learning many levels of representation and abstraction in order to inspire recognition of data, for example text and images (LeCun & Ranzato, 2013). Which are smart, implementable and have the capability to fulfil performance requirements for a number of practical applications, the computers performance is still far behind that of a human.

The neural network models are simple when comparing neural networks with complexity of the human brain that includes billion of neurons, where each one is connected to many of neurons (Johnston, 2008). HTM theory was developed by Jeff Hawkins and Dileep George and with great interest by Hawkins and his group at Numenta. (Hawkins *et al.*, 2011) Jeff Hawkins proposed HTM which is not a new theory but collects existing ideas. He developed it in order to mimic the neocortex, a model of neocortical computation. This theory is so close to the findings from neuroscience (Byrne, 2015). The Hierarchical Temporal Memory Cortical Learning Algorithm (HTM CLA) is the novel path of approaching problems in Machine Learning and Artificial Intelligence. HTM CLA gave support in the utilisation of knowledge of cortical structure and machine learning technology to solve the problems in this area. The HTM network is a new form of neural network where HTM is member in a family of biologically inspired machine learning algorithms such as Convolutional Neural Networks and Deep Belief Networks (Price, 2011). The author focuses on the knowledge of biological intelligence to build smart machines. Although Neuroscience illustrates the behaviour of neurons and the functioning of the brain, however, our knowledge about our neuronal circuitry is still limited. The HTM CLA is a new approach that attempts to fill the gap between neural mechanisms and intelligent behaviour (Price, 2011).
In this chapter, this research introduces HTM theory that presents how the neocortex works and explains the neuroscience of how the brain biologically works. It illustrates how an ANN works that attempts to mimic the human brain.

3.2 Biological Neural Network

The human brain consists of around 10 billion biological neurons in the human cortex that connect with each other through synapses. The number of synapses in a single neuron may reach a thousand synapses. The human brain is able to receives the signal, process it, and send it by the axon in the nervous system. The biological neural network is composed of the soma (cell body), the axon, dendrites, and synapses. Axons connect to dendrites by synapses, and dendrites interact with their neighbours or other neurons through the synapses as shown in figure 3.1. When the total of the input signals (action potential) into one neuron exceeds a threshold, it will fire a neuron. During the learning phase, a synapse may increase (excite) or decrease (inhibit) action potential. This connection between neurons gives them the ability to store information, images, sound, and other signals that reach them through the five senses. This allows them to learn through repetition and from errors. Neuroscientists have made progress on how the human brain works in terms of learning, remembering, recognising objects, and making decisions. A hierarchal system is one of the discoveries that gives the human brain the

ability to process input data. Each region has a specific functionality, and the higher we go in the hierarchy, the higher the abstraction (Zurada, 1992).

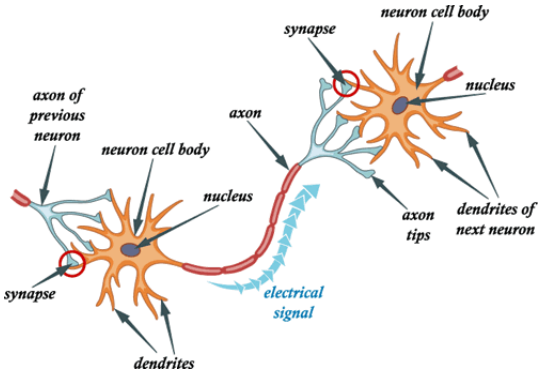


FIGURE 3.1: The biological neural network (Devineni, 2015).

3.3 History of Neural Networks

(McCulloch & Pitts, 1943) conducted a study about how the brain works and tried to recognise the ability of the brain to produce highly complex patterns using many neurons that are connected. It has been simulated using simple electronic circuits. They made a model of a neuron, known as the McCulloch and Pitts model (MCP neuron), and it contributed to the development of Artificial Neural Networks (ANNs). (Hebb, 1949) explained that the synapses between the neurons become strong when used more. It means that there are a number of neurons next to each other but two of them are moving the data intensively. It forms strong lateral connections between the cells that process and transmit neural activation. In the future, when given the same input, those cells will be excited. (Rosenblatt, 1958) proposed the concept of the perceptron that passes inputs by pre-processors, which are called association units that are pattern detectors, and the perceptron is a pattern recognition device. (Minsky & Papert, 1969) published a book that illustrated there was a limited of one perceptron, it cannot perform perfectly for exclusive OR (XOR) operations. In 1986, multilayer perceptrons were the solution to this problem, through the extension of the single-layer architecture to use an extra layer called a hidden layer. (Eluyode & Akomolafe, 2013) conducted a comparative study between biological and artificial neural networks regarding how to combine a computer and brain. The computer can effectively compute arithmetic and logic operations. However, the brain is able to learn and recognise patterns of variation (Chen *et al.*, 1986). There are several computing manufacturers that have tried to produce a machine that can process data as patterns, similar to the work of the human brain. (Eluyode & Akomolafe, 2013) evaluated the advantages of biological and artificial neural networks to compare them and to determine the features of the biological

TABLE 3.1: Analogy between biological and ANN neural-networks (Eluyode & Akomolafe, 2013).

Biological Neural Network	Artificial Neural Network
Soma	Unit
Dendrites	Input
Axon	Output
Synapse	Weight
Potential	Weighted sum
Threshold	Bias weight
Signal	Activation

neural network, which can be adapted for creating an intelligent machine. The authors focused on neural network criteria, for example structures, hidden layers, processing units, etc. There are many components in biological neural networks. However, artificial neural networks also include several components that are equivalent, as shown in Table 3.1. They illustrated that biological neural networks have many neurons, and the interconnection method between neurons is absolute. Dendrites send input signals to the cells, and a synapse is placed to store the information. Through the learning step, a synapse may increase or decrease the action potential. After the cell produces a signal, the axon is able to convey an output signal to another cell. In artificial neural networks, the grouping comes through building layers, and the connections pass signals from one neuron to another, where each unit receives many inputs and produces output. It is similar to the axon in the biological neuron, and learning occurs by adjusting the connection weights between units, similarly to what happens in the synapses. The axon forwards a signal that reproduces from cell to cell, and the other cell receives the signal as an input that the axon combines with the dendrites near the cell. While artificial neural networks focus on the weights and adjusting the connection weights between units, they do not focus on distal dendrites.

3.4 HTM: Overview

The HTM is a special approach to machine-learning techniques, which begins from the neuroscience of the neocortex. The focus of the theory is on the neocortex because it is envisaged to be where human intelligence resides, and the theory is attempting to learn how intelligence works in it. Additionally, the purpose of studying the neocortex from the computational side is to build, design, and implement machine-learning models.
The neocortex is the sheet covering the higher brain. The size of the neocortex is around 2.5mm thick and contains billions of brain cells or neurons. There are parts under the neocortex that are engaged in the essential tasks of life, such as sleeping, eating, etc.

However, these activities do not represent any real intelligence. In the neocortex, the images, information, and everything received from the sensors are stored. The neocortex is able to learn and remember and recognise patterns in life. The neocortex is the location of human intelligence in the brain. The neocortex is split into many regions. The regions connect with other regions in a hierarchical structure. Each region receives the raw sensory data from the lower region and passes it up to higher regions.
The regions contain many cellular layers (slices), and the structure of each one of these layers is almost identical. Every region of the neocortex, such as the auditory and visual regions of the brain, have the same structure. This shows that the brain performs the same cognitive operations across all regions. This common operation tries to reason and understand the information coming from the sensors. The recognition is different in the higher regions compared to the lower regions. The lower regions of the hierarchy learn primitive objects. For example, in the visual system, the lower regions recognise primitive shapes and edges of the observed image. The higher in the hierarchy, the more sophisticated and abstract these recognised objects become. In the hierarchy, there is a correlation of ideas, activities, other memories, and some ideas or information stored in the neocortex that are understood over time and become more abstract. There are collaborations between these regions to achieve certain cognitive goals.
The HTM is a memory system based on recalling previously learned sequences of memories. At every time step, the HTM model learns and updates its beliefs about the world and adapts to this new information. From these historical memories, the HTM model is able to predict future events or patterns. Recently, ANNs, convolutional neural networks, recurrent neural networks, and deep learning have achieved great successes, but they are not a faithful biological modelling of how the neocortex works.
The ANN includes one or multiple hidden layers and one input and one output layer. Each layer is a collection of nodes representing the neurons. Each node receives its input from connections from the previous layer. The connection between these nodes has a certain weight. The output of a node is a weighted sum of the connected synapses from the previous layer. In ANNs, the node is called a point neuron, as shown in Figure 3.2 (a).
There are many types of neurons in the neocortex. The most common biological type is the pyramidal neuron, as shown in Figure 3.2 (b). Every pyramidal neuron has several thousands of synapses distributed across their dendrites. Through the learning phase in a biological neuron, the synapse can be excited and form strong connections, or the connections can be weakened, and the synapses will disconnect.
The pyramidal neuron has three sources of input. First, input coming from the proximal dendrites represent the feed-forward input. Second, input coming from the distal dendrites form lateral connections to neighbouring cells. Third, input coming from apical dendrites receives the feedback from higher regions as shown in Figure 3.2 (c).

The HTM artificial neuron usually has many synapses that form connections to the dendrites, similar to the pyramidal neuron as shown in Figure 3.2 (c). During the learning phase, the synapse connections can be strengthened or weakened. This process is the essence of learning in the typical ANNs. However, in the HTM, each synapse has binary states that can be either connected or disconnected. What determines the state of the synapse is the strength of the connection, which is a scalar value (similar to the weights in ANNs). If the strength exceeds a certain threshold, the synapse is said to be connected, otherwise it is disconnected. The HTM neurons attempt to model a more realistic representation of the real neurons in the brain than the point neurons (Hawkins *et al.*, 2011), (Hawkins *et al.*, 2016),(Hawkins & Blakeslee, 2004b).

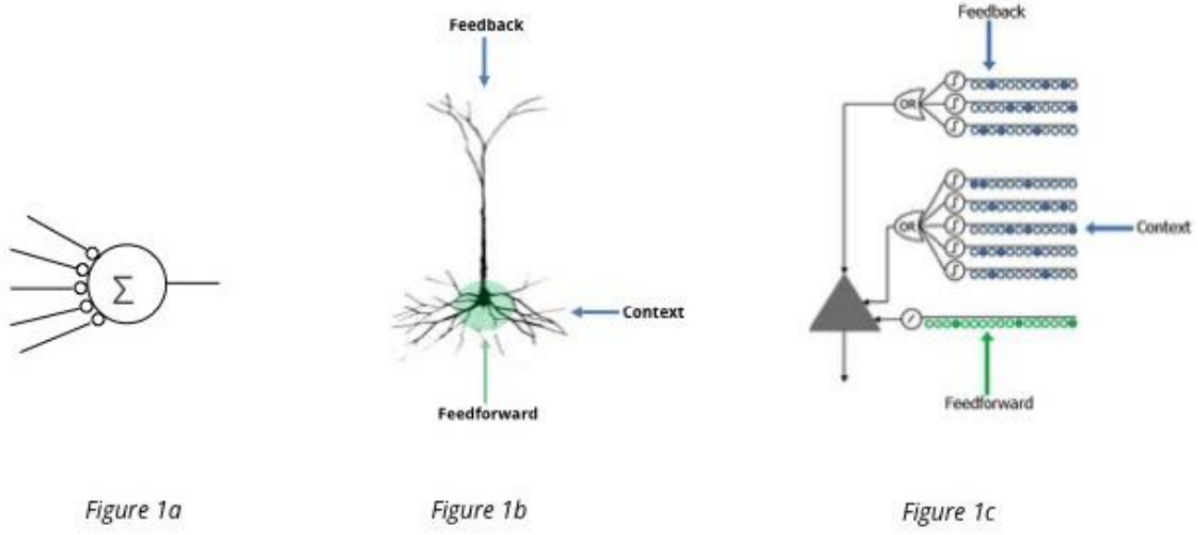


FIGURE 3.2: Biological and artificial neurons (Hawkins *et al.*, 2016).

3.4.1 HTM Concept

HTM was developed at the Numenta Association (Hawkins *et al.*, 2011) (George, 2008). HTM contains a hierarchical architecture as mentioned in (Hawkins & Blakeslee, 2004b). Jeff Hawkins developed it in order to mimic the neocortex. HTM contains a hierarchical architecture. HTM has many regions and those regions are connected together through elements. The lower level receives its inputs through a small region that comes from the senses. The lower level of output is passed up to the higher levels of the hierarchy to match receptive fields (Hawkins *et al.*, 2011).

Programming HTM, is very different from programming traditional computers because HTM is trained by a stream of sensory data as it is a memory-based system. HTM networks are trained many times on varying data, storing many patterns and sequences. In HTM, data storage and access are logically different from standard models that have

been used for programming up till now. This is because classic computer memory has a flat organisation that enables it to carry out every kind of data organisation and structure, and programmers can control the storing of information. In HTM, however, memory is restrictive and organised hierarchically, so its memory is inherently time-based where the information is stored in a distributed way. Although their main functions are based on hierarchy, time and sparse distributed representations, unlike in classic computing, HTM networks can be used as general purpose computers (Hawkins *et al.*, 2011). HTM networks contain regions sorted into a hierarchy where the regions are considered as important parts of memory and prediction. In the hierarchy, each region represents one level. Each level has a region, and each region has many elements, with many elements connected to parents although elements are at different levels. This allows the integration of many HTM networks that have more than one sensor or source. In order to explain in more detail about levels and regions, a region refers to the internal function of a region while the level refers specifically to the role of the region within the hierarchy. Hierarchies have many benefits, including decreased training time and memory usage. It can be a simpler solution for prediction problems with a single HTM region. The concept of regions wired in a hierarchy and the connection of regions with each other mimics the biology of the human brain, the regions can either receive inputs directly from the senses or after they have passed many regions (George, 2008), (Hawkins *et al.*, 2011).

3.4.2 Hierarchy

The HTM networks contain regions arranged in a hierarchy, in which each region combines a spatial pooler and temporal memory that have hundreds of cells arranged in columns. Each region has many elements that are connected to other regions through elements at different levels. The input comes from lower regions as feed-forward input, or feedback is received from higher regions. In HTM, however, memory is organised hierarchically to reduce training time and memory usage.

3.4.3 Sparse Distributed Representations (SDRs)

The neocortex has many neurons that can be interconnected, but only a small number of the neurons are active at one time: this is called encoding, in “sparse distributed representation”, “sparse” means that only a small percentage of neurons are active at one time. The active bits usually constitute around 2% of the whole SDR array. “Distributed” means that the activation of many neurons is required in order to represent something

meaningful (Hawkins *et al.*, 2011). The neocortex has many neurons, each region represents a small number of the neurons that are active at one time, from which the bits have semantic meaning, called Spares Distributed Representation (SDR). Each SDR has a number of active bits representing a semantic meaning. When two SDRs in the exact same position are active bits, called the overlap, the two SDRs have similar semantic meanings (Hawkins *et al.*, 2016).

The SDR is the fundamental information representation in the CLA. This point offers mathematical definitions and notations. (Ahmad & Hawkins, 2015) presented several definitions and mathematical notations:

- Binary arrays (SDR): The SDR is an array that includes a set of zeros and a small percentage of ones. The size of an array is denoted by n, where n is indicated as a fixed total number of bits. In an SDR, x contains an n-length vector of binary data, and i refers to the index location of a number in an array, such as $x = [b_0, b_1, \ldots, b_{n-1}]$.

- Overlap: Overlap is an operation to identify the similarity between two SDRs using an overlap score. The overlap score is determined by the number of active bits overlapping between two SDRs, when two SDRs in the exact same position are active bits. The overlap score may be calculated as the dot product when there are two SDRs, such as x and y:

$$overlap(x, y) = x \cdot y \tag{3.1}$$

- Matching: To ensure two SDRs are matching, but not an exact match as mentioned earlier regarding the overlap, the parameter θ is used, which denotes the threshold to determine a match between two SDRs. When the overlap score exceeds the threshold, it is either a match or not an exact match. For two SDRs x and y:

$$match(x, y) = overlap(x, y) \geq \theta. \tag{3.2}$$

  We can determine an exact match by setting . To determine a true match through tuning , the parameter should be less than or equal to the active bits.

- Sparsity: The sparsity is a percentage of (the number of active bits to the total number of bits), where w refers to the number of active bits. This equation is indicated by s; for example, for SDR x, the sparsity can be computed using the equation below:

$$s(x) = \frac{w}{n} \tag{3.3}$$

3.4.4 Time

HTM theory argues that time is very important in learning, inference and prediction, so all HTM systems obtain time changing inputs through training. Time plays an important role in how the brain actually understands the world, according to the HTM theory. When the brain receives the sensory input patterns that are captured from an object, it needs time in conjunction with the stream of the sensory input to recognise the object. Without this time-changing input of the object, the brain cannot recognise the object. HTM needs to recognize these patterns and train on data that have been received from sensors over time (Hawkins *et al.*, 2011).

3.5 Cortical Learning Algorithm (CLA)

The cortical learning algorithm (CLA) is the computational implementation of HTM theory that attempts to mimic the neocortex. The CLA model has been tested and implemented in software by Numenta‘s project Nupic. The CLA is an algorithm that performs the operation of a single region. This region includes the spatial pooler and the temporal memory, which are two significant components of this algorithm. The CLA’s main features include the ability to learn from every cell in the input space and its relationship to each column, where each column represents the semantic meaning. Each cell in that column represents that same meaning but in a different context and predicts temporal patterns (Hawkins *et al.*, 2011), (Agrawal & Franklin, 2014).

3.5.1 Understanding How the Cortical Learning Algorithm Works

The CLA is a machine-learning algorithm that is based on HTM theory, which aims to explain the structural and algorithmic characteristics of the neocortex (Hawkins & Ahmad, 2016). A typical CLA model consists of four main components. The first component is the encoding component, which can contain one or more encoders that read the input data and convert them to an SDR. The next component is the spatial pooler, which receives the outputted SDRs from the encoder. The spatial pooler learns the spatial features of the passed SDR and creates another SDR and output it to the next component. The third component is temporal memory, which learns the temporal changes in the SDRs. Finally, for prediction and classification problems, a CLA classifier component sits at the top of the CLA model. The CLA/SDR classifier component decodes the state of the CLA model and produces predictions (Hawkins *et al.*, 2016). Figure 3.3 shows the process flow of the NuPIC framework .

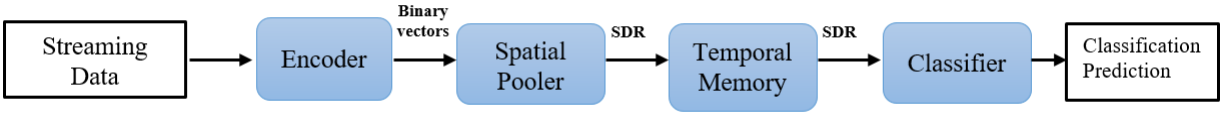


FIGURE 3.3: Work flow of an HTM application.

3.5.2 Cell States

The cells (neurons) in the CLA region can be in one of three different states: active, inactive, and predictive. The active state refers to the cells that can become active when signals are received from proximal dendrites. Second, the inactive state refers to the cells that have not received sufficient signals from proximal dendrites or distal dendrites. The third is the predictive state that receives sufficient signals from the distal dendrites.

3.5.3 How the Cortical Learning Algorithm Learns

Learning occurs by changing the effectiveness of the synapses, which affects connections between neurons. Synapses have a binary on/off state, and the synapses have permanence that is a floating-point number from 0.0 to 1.0 and represents the strength or weaknesses of the connection. If the strength of the permanence exceeds a threshold, the synapse will be connected. In the CLA, the cell contains two inputs, input from the proximal dendrites that represent the feed-forward input received from sensory input or outputs of the lower-level region, as shown in Figure 3.2 (c). The green is a proximal dendrite. The second input is received from the distal dendrites that form lateral connections to neighbouring cells that belong to the same CLA region. As shown in Figure 3.2 (c), the blue denotes many distal dendrites. All dendrites of a cell have many synapses (Hawkins *et al.*, 2011).

3.6 Components of CLA

3.6.1 Encoder

An encoder receives the a sensory's data type and converts it to an (SDRs), which will be fed into an HTM, similar to an ASCII code representing the data on a computer. The encoder extracts the output that consists of ones and zeros. NuPIC, the open source project, has already used sets of encoders such as Scaler encoder, date encoder and

category encoder, etc. The encoder selection depends on the data type. When the sensory input has semantic meaning, similar data must output extremely overlapping SDRs (Purdy, 2016).

3.6.1.1 Characteristics of Encoders

(Purdy, 2016) offered how to represent data as SDRs in NuPIC. Although there are many encoders available, the NuPIC does not fulfil the requirements for most applications. However, in order to create a new encoder, the following rules should be considered when encoding data:

1. **Semantic meaning**: Any two similar sensory inputs should have overlapping active bits.
2. **Deterministic**: Each same input must always take the same resulting SDR as output.
3. **Fixed-in dimensions**: The output SDRs should have a fixed total number of bits.
4. **Fixed-in sparsity**: The number of active bits should be constant for the resulting SDRs.

3.6.2 NuPIC Encoders

3.6.2.1 Encoding Numbers (Scalar Encoder)

Standard NuPIC includes many encoders that can deal with numerical data types. The scalar encoder can be utilised to encode integers and floating-point number. The figure 3.4 represents 2D SDRs with certain parameters set by a user to determine the total number of bits, minimum value (1), maximum value (40). Any number larger than 40 gets the same encoding. Unless the periodic flag is set, the active bits will cycle and loop from the beginning of the allocated space. and The number of buckets can be computed by this equation $n-w+1$. The buckets enable the user to determine the total number of bits (n) and the number of active bits indicated by (w) to encode the data. In this figure 3.4, There are overlapping bits between the number 1 and the number 2 (bit at index (1) and the bit at index (2), staring the index with zero). The higher the similarity is, the higher the number of active overlapping bits.

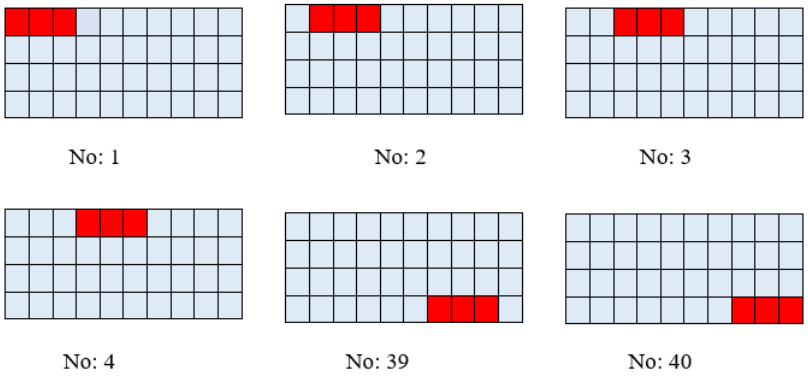


FIGURE 3.4: The Scalar Encoder $[1, 2, 3, 4.., 40]$ using encoding the parameters $n = 40, w = 3, minVal = 1, maxVal = 40$.

TABLE 3.2: Represents the number of buckets.

1	$\rightarrow$	[1 1 1 0 0 0 0 0 0 0]
2	$\rightarrow$	[0 1 1 1 0 0 0 0 0 0]
3	$\rightarrow$	[0 0 1 1 1 0 0 0 0 0]
4	$\rightarrow$	[0 0 1 1 1 0 0 0 0 0]
5	$\rightarrow$	[0 0 0 1 1 1 0 0 0 0]
6	$\rightarrow$	[0 0 0 0 1 1 1 0 0 0]
7	$\rightarrow$	[0 0 0 0 0 1 1 1 0 0]
8	$\rightarrow$	[0 0 0 0 0 1 1 1 0 0]
9	$\rightarrow$	[0 0 0 0 0 0 1 1 1 0]
10	$\rightarrow$	[0 0 0 0 0 0 0 1 1 1]

The buckets are the window of active bits for a scalar encoder with n=10, w=3. Table 3.2 above shows each number with its SDR. It is obvious that the number of unique SDRs is 8, which represents the number of buckets :

n =10, w=3, buckets= 8.

n - w + 1.

buckets= 10 - 3 +1 = 8.

calculate the total number of bits,

n = buckets + w - 1

n = 8 +3 - 1 =10

The scalar encoder has many interfaces to determine the total number of bits, and, in each way, maintains the required encoder properties. The resolution parameter can

replace the parameter (n) if the *resolution* of the encoder is important to the user. For example, when `resolution=1, w=1, minVal=1, maxVal=10`, the encoder will be able to encode ten unique SDRs. If the *resolution=2*, the encoder will be able to encode twenty unique SDRs. In other words, if the user wants to encode integer values from 1 to 10, the user can set the *resolution=1* which will give every integer one unique representation. If the user wants to encode decimal values, the resolution should be increased. For example, if `resolution=2, w=1, minVal=1, maxVal=10`, the encoder will be able to distinguish between the number 1.0 and the number 1.5 as each one will have a slightly different SDR. The higher the resolution is, more representations are available to encode small decimal values. The scalar encoder is facing a problem with some numeric data, if the minimum and maximum values are not known in the data. This is because the scalar encoder requires the user to identify certain minimum and maximum value to be able to allocate the required number of bits. Hence, it is not possible to make any changes through the learning process. Allowing the use of the adaptive scalar encoder that adapts to the users changing parameter settings. The adaptive scalar encoder is an improvement over the regular scalar encoder and does not require the minimum and maximum values to initialise the encoder. It adapts to the data that it receives by tracking the minimum and maximum values. It is noted that the adaptive scalar encoder does not support a resolution parameter. In order to clarify the way it works an adaptive scalar encoder through pass the encoder the scalar values (1, 2, . . , 10) two times and presents the resulting SDRs. Table 3.3 below shows the first pass. When receiving the first number the encoder will make it the minimum value, when receiving another value bigger than first value the encoder will make it the maximum value.

TABLE 3.3: Adaptive Scalar Encoder through first pass.

1	→	[1 1 1 0 0 0 0 0 0 0]
2	→	[0 0 0 0 0 0 0 1 1 1]
3	→	[0 0 0 0 0 0 0 1 1 1]
4	→	[0 0 0 0 0 0 0 1 1 1]
5	→	[0 0 0 0 0 0 0 1 1 1]
6	→	[0 0 0 0 0 0 0 1 1 1]
7	→	[0 0 0 0 0 0 0 1 1 1]
8	→	[0 0 0 0 0 0 0 1 1 1]
9	→	[0 0 0 0 0 0 0 1 1 1]
10	→	[0 0 0 0 0 0 0 1 1 1]

And now the second pass values (1, 2, . . . , 10):

```
1   →  [1 1 1 0 0 0 0 0 0 0]
2   →  [0 1 1 1 0 0 0 0 0 0]
3   →  [0 0 1 1 1 0 0 0 0 0]
4   →  [0 0 1 1 1 0 0 0 0 0]
5   →  [0 0 0 1 1 1 0 0 0 0]
6   →  [0 0 0 0 1 1 1 0 0 0]
7   →  [0 0 0 0 0 1 1 1 0 0]
8   →  [0 0 0 0 0 1 1 1 0 0]
9   →  [0 0 0 0 0 0 1 1 1 0]
10  →  [0 0 0 0 0 0 0 1 1 1]
```

When receiving the first number the encoder will make it the minimum value, when receiving another value, the encoder will shift to the left and creates a sliding window of active bits, which include the range from a set of values.

3.6.2.2 Category Encoder

A categorical encoder is used to encode categorical data inputs, as shown in figure 3.5, such as cat, dogs, etc. Sometime the data contains discrete data, unrelated or related (such as days of a week). The encoding in these situations should try to reduce the overlap among the category encodings. To determine a few numbers of bits for each option, the encoding of each option has active bits, for example of the weekday/weekend encoding. This encoding is very important to deal with data streams; there are the various input patterns between weekday and weekends that make it easy for the HTM systems to recognise and learn separate predictions for the two periods.

Figure 3.5 shows three categorical values (cat, dog, bird) and the resulting SDRs after encoding them with the category encoder. It is worth noting that the first bucket is allocated for any unknown categorical value other than the known values (cat, dog, bird) with the following parameters: `w=3`.

Category	SDRs
CAT	000111000000
DOG	000000111000
BIRD	000000000111

FIGURE 3.5: The Category Encoder.

3.6.2.3 SDR Category Encoder

Similar to the scalar encoder, the category encoder faces an issue when the categorical values are not known. This is especially the case in online prediction. The NuPIC develops the SDR category encoder to overcome this problem of knowing all values that can be passed to the encoder. The SDR category encoder work by assigning each categorical value a random and distributed number of active bits while trying to reduce the probability of having overlaps between the resulting SDRs.

3.6.2.4 The Date Encoder

The date encoder is similar to a scalar encoder with the periodic flag turned on. Actually, it relies on the scalar encoder and it just provides a better interface for dealing with dates. There are many parameters that will be determined through the date encoder:

- `timeOfday`:Which specifies the time of the day in hours.
- `DayOfWeek`:Which specifies which day it is (Monday = 0 to Sunday = 6).
- `Weekend`:Which is a Boolean value (true or false). Whether the date is a weekend or not, it will be on or off. Figure 3.6 shows the date encoder encoding a date and time value using the season parameter. The first input is in May and the second one is in June. Therefore, there are overlapping between the two SDRs. Unlike the last input which is in December which has no overlapping bits with the first two inputs.

Date/Time	SDRs
2018-04-10 : 20.15 Pm	00001111100000000000
2018-05-10 : 20.15 Pm	00000011111000000000
2018-12-10 : 20.15 Pm	11000000000000000111

FIGURE 3.6: The Date Encoder.

3.6.3 Spatial Pooler

The spatial pooler is the second component of HTM/CLA. The base function of the spatial pooler is to receive an input (as input space) from the encoder and transform it to an

SDR. The input space has many input bits. To maintain a fixed sparsity for these inputs, the spatial pooler maintains a specific sparsity to maintain the semantic meaning. The spatial pooler is able to deal with input space and convert it into an output vector of a different size with sparse active bits within a fixed number of active mini-columns as shown in figure 3.7. The spatial pooler learns from every cell in the input space and from its relationship to each column. Each column contains a number of proximal dendritic segments, which is a set of potential synapse connections to a subset of the input bits, determining a subset of the input space that is connected to each column according to the synapses that have permanence value. When the permanence value is above the threshold, the synapses will be connected. Each column includes potential connections (cells) from input space within a certain radius around it. It does not include all input space. Some cells of the input space are connected to the active column. The overlap score of each column is activated by a number of synapses on each column connected to active input bits in the input space.

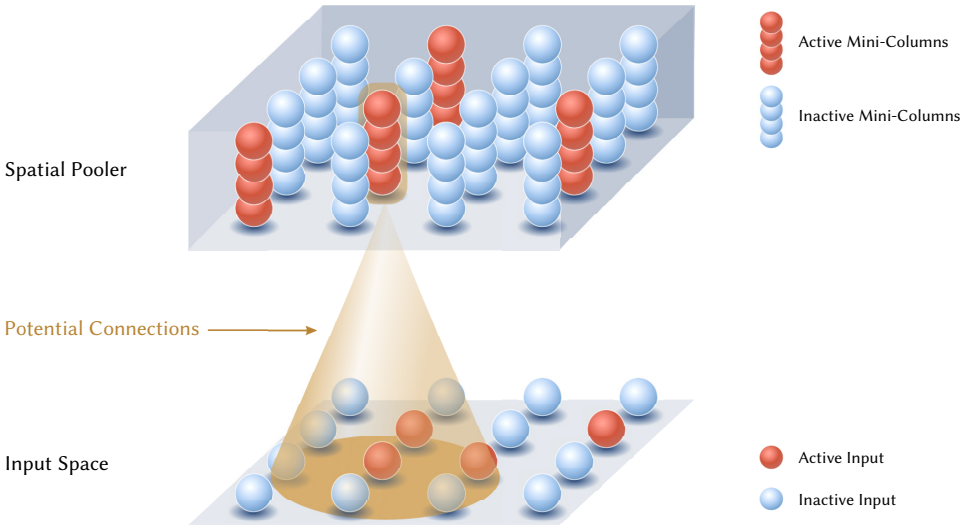


FIGURE 3.7: A mini-column in the SP and its receptive field (Alshammari, 2018).

The spatial pooler requires a specific number of columns that should be set in pre-training the CLA model. The spatial pooler exploits these columns to contribute to the learning process. The spatial pooler uses a boosting method, which attempts to enhance the losing column to be more powerful and compete to learn the coming inputs. Boosting is used to split up various patterns, which share a minimum of active columns. It enables the temporal memory to distinguish various patterns. Each synapse is connected, and the synapses will be activated. The number of active synapses is multiplied by a boosting factor. This boosting factor is specified through how many times a column is active. After boosting, the column will be active with the highest number of active synapses. The winning column is based on a threshold that is set by the user. When

the columns are winning with the highest overlap as active, it will inhibit their neighbouring columns from winning; thus, a small percentage of columns are active in this time. During the learning, many permanence values that will increment and decrement will be updated. Depending on when the synapses are connected to the active bits in the input space, permanence will be increased. When the synapses are connected, the inactive bits are decreased (Hawkins *et al.*, 2016) (Cui *et al.*, 2017).

3.6.4 Temporal Memory

Temporal memory is the third component of HTM/CLA. The temporal memory learns SDRs generated by the spatial pooler and makes prediction patterns next. It illustrates how cells (neurons) in the neocortex can remember spatial sequences in the context of prior inputs by activating particular cells in each column. Temporal memory performs two tasks: 1) learn temporal changes in the SDRs generated through SP over time and 2) predict future patterns on the basis of the temporal context of each input. The temporal memory includes three phases. The first phase determines which cells in the active columns will become active. When a column becomes active, this is because it looks at all the cells within each active column to feed-forward input. When none are in the predictive state, it will activate all cells in this column, which is called bursting. When each cell in the active column is already in the predictive state, only those cells become active. In the first step in the sequence, all cells are active within each active column because they have never seen the input pattern in a context before. The current input pattern matches only the activating cells in the predictive state. If the input pattern is expected, only those cells will become active, or all cells in the column will activate when the input pattern is unexpected. In the second phase, select cells become a predictive state. The distal input forms lateral connections to other cells. The cell segments in the predictive state form lateral connections to the cells that are active in the current time step and go through the dendrite segment and count all active synapses, and those above the threshold become predictive. All the cells in the columns may be in a predictive state, going through the dendrite segment and calculating all the connected synapses to active cells. When the number of active synapses exceeds the threshold, the segment becomes active. All the cells with active dendrite segments are put into a predictive state. We have two types of dendrite segments, proximal and distal. Synapses connect to dendrite segments. The proximal dendritic is the feed-forward input from the input space. Each HTM cell has one proximal dendrite, when a column becomes active due to receiving a signal from the proximal dendrites. In the third phase, the synapses are updated. All the synapses have permanence values that are modified on the basis of the segment. When the distal dendrite segment becomes active, the increment or decrement

of the permanence of those synapses is according to active cells or inactive cells. When the cell correctly predicts the feed-forward input, the permanence of those synapses for the segments is incremented and the wrong predictions are penalised (Hawkins *et al.*, 2016).

3.6.5 Classifier

The last component of the HTM model is called the Classifier which decodes the state of the HTM model and produces predictions. Data comes in scalars, categories, data/time, etc. The encoder receives the data and converts it to SDR. Then, it passes the SDR to the SP and then to the TM. The model can be used with typical architecture of applications such as to perform prediction, anomaly detection, and sequence classification. The classifier is needed to perform prediction problems. Many regions such as SDR Classifier, CLA Classifier, KNN Classifier, and Anomaly scores/likelihood can be added to the HTM model in order to perform the required application. These regions are not biologically inspired. The current implementation of the CLA in the NuPIC framework has several classifiers: the k-nearest neighbour (KNN) classifier, CLA classifier, and the SDR classifier which are presented in the following sub-sections.

3.6.5.1 K-nearest Neighbours Classifier

The KNN classifier is a predictive model used for classification. Most predictive models are focused on datasets to determine data patterns, but the KNN ignores some data and predicts a new instance based only on the number of nearest neighbours of the point. An element is classified by a majority vote of the nearest neighbours measured by Euclidean, Manhattan, and Minkowski distances so that the distance value is calculated among a new example with all examples in the dataset using distance functions. The implemented NuPIC KNN classifier has some advantages, such as an optimised C++ class. It is suitable for sparse vectors and able to operate in an online fashion. Moreover, it provides different distance methods (e.g. Euclidean, Manhattan, and Minkowski) (Numenta KNN Classifier, 2017). Within NuPIC, there is an implementation of the KNN algorithm, and the it has been applied in many domains, such as vision problems and classification of images (Fallas-Moya, 2015), as well as object tracking based on hierarchical temporal memory (HTM) classification. The KNN technique is not biologically inspired. The KNN classifier includes many parameters, such as the following:

- **k**: is an integer value specified by the user and must be an odd number that is used to determine the value of the variable k, which expresses the number of nearest neighbours for each point.
- **rawOverlap**: is suitable for binary input.
- **Norm**: determines the kind of distance functions, such as Euclidean, Manhattan, and Minkowski, to measure the distance between classified patterns and a new pattern proposed to be classified.

3.6.5.2 CLA Classifier

The CLA classifier is necessary to decode the state of the CLA model and produce predictions. The CLA classifier receives the SDR output from the TM (active cells), Predicted Field (PF), bucket index, and record number from the encoder. It seeks to learn a function of this information at that time step, and it generates a probability distribution for the PF. The label with the highest associated probability is considered the output.
The CLA classifier receives the SDR output from the temporal memory (the activation pattern) and the target input from the encoders describing the input to the system at that time step. In the learning phase, the history of the classification is recorded for every bit in the activation pattern. The history is weighted, where the latest activity has more effect than the older activity, and the alpha parameter manages this weighting. In the inference phase, every active bit in the activation pattern is checked. It searches for the most likely classification based on the history stored for that bit, the probability distribution over the PF is determined, and the votes on these to obtain the output.

As shown in Figure 3.8, all fields are passed to the encoder and converted to an SDR, the CLA Classifier also receives the ground truth labels to learn from. The CLA Classifier also receive the `bucket index` and `Record number` . The CLA Classifier output is a probability distributions of the predicted labels and the label with the highest associated probability is considered the output.
We assume that streaming data are fed to the system. Most machine algorithms are set up as spatial problems that predict values at the same time (t). When comparing with the CLA Classifier it is predicting a future value at time $t + k$ (where k is the predicted number of steps into the future). Other problems to consider are that the dataset could be split into training data and test data. In the training phase, the predicted field is given to the algorithm, but the testing data does not give this algorithm a predicted field directly (a ground truth label) to the algorithm, for it to do the classification. The CLA Classifier does not resemble the biological function of the human brain. Although it is able to predict next time step. In order to work the same way as a human brain, there

is a need for many hierarchies of various modalities. (Balasubramaniam *et al.*, 2015), (Numenta CLA Classifier, 2017), (Classifiers NuPIC, 2017).

The CLA classifier contains many parameters:

- **Alpha**: is a parameter that serves to change the output parameter to identify how far back to remind. It is applied in predictions. When the prediction is a categorical value, there is no need to use the moving average.
- **The parameter steps**: contains the number of steps into the future that the classifier predicts, which is determined by the user.

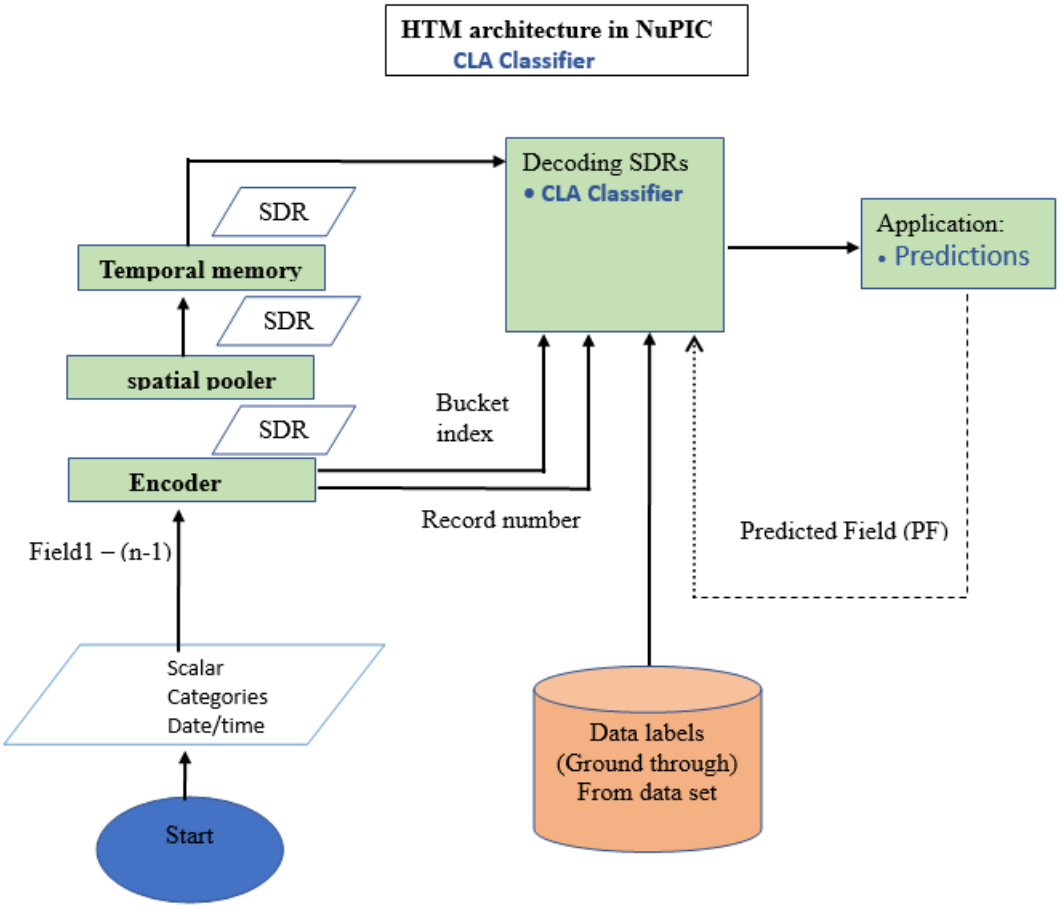


FIGURE 3.8: A CLA Classifier workflow.

3.6.5.3 SDR Classifier

The SDR classifier also receives the SDR output from the temporal memory and the target input from the encoders. It maps the input SDR to the class labels and produces a

probabilistic distribution over all class labels. In the inference phase, a weighted summation of all the inputs are calculated, and then the output is passed to a softmax nonlinear function to determine a probability distribution of class labels. In the learning phases, during the training of the data, the output units are checked. When the results are not satisfactory, there is a need to adjust the weights to reduce the error rate and maximise the likelihood of the model.
The aim of the SDR classifier is similar to the CLA classifier. It attempts to infer the output from SDR (active cells in TM) in the top region in the hierarchy. The SDR classifier learns among active cells in TM at time t, and the PF is fed at time $t + k$ (where k is the number of steps into the future to predict), the record number and bucket index from the encoder. The SDR classifier maps the SDR (active cells) to class labels. The output is a probabilistic distribution on all classes. The predicted labels and the label with the highest associated probability are considered the output.

The SDR classifier applies a single layer, the feedforward classification neural network. The network has input units that comprise the array of TM active cells (0, 1) and output units.

The SDR classifier includes many parameters:

- **Steps**: The step is a parameter that is identified by the user. This is how many steps to predict the value of this predicted field into the future.
- **Alpha**: Alpha is a hyper-parameter used to adjust the weights through the learning phase. The lower the Alpha, the slower the learning rate goes along the downward slope. The larger the value, the faster adaptation to the data[1], (Classifiers NuPIC, 2017), (Yuwei, 2017)[2].

The properties of the SDR classifier derived from HTM are as follows:

- The HTM has the ability to learn from a streaming data in an online fashion that adapts to the users changing parameter settings.
- The HTM is able to deal with multiple predictions simultaneously. In the learning phase, it estimates the full predicted distribution at each step. It reinforces correct prediction sequences and penalise the wrong prediction.
- In HTM, the TM output is SDR that contains ones and zeros. Each active cell that has a value of 1 needs a weight, and there are sets of weights in a matrix that update these weight values at each step to avoid overfitting.

[1]http://nupic.docs.numenta.org/1.0.3/api/algorithms/classifiers.html
[2]https://github.com/numenta/nupic/pull/3066

3.7 Planning for the CLA

The typical process for creating CLA models starts with a parameter optimisation step. This step is called swarming in the CLA, which uses Particle Swarm Optimisation (PSO), to assess the model performance for the given task. In this step, the model will be executed on a dataset and search the parameter space and optimise for the best parameters for that model. Then, the second step is carried out which is the training phase of the model. Finally, the model is tested on a portion of the dataset that it has never seen before and the performance of the model is calculated afterwards (Lee & Rajabi, 2014). The whole process is facilitated by the Online Prediction Framework (OPF), which is a software framework developed by Numenta to streamline the model creation, optimisation, and evaluation.

3.8 Particle Swarm Optimisation (PSO)

Swarming is used to set the parameters of the model on the dataset. Swarming is a procedure that selects the best model of several options for a given dataset. The swarm process can assess several models and produce the best model based on the lowest error rate on the given dataset. There are many components in NuPIC that require parameter values to configure them. Swarming provides the best parameters to tune every component, such as encoders, spatial poolers, temporal memory, and classifiers. It attempts to evaluate a number of parameters. There is an extremely large search space; thus, swarming uses efficient methods and intelligently reduces the search space to gain the best performance model and reduce time[3].

3.9 Summary

In this chapter it was defined how neurons in the neocortex work, explained how the human brain works by combining neuroscience and computing findings that enable a modern understanding of intelligence itself, which assists in creating intelligent machines. The Chapter covered the concept of the HTM theory and its CLA implementation that includes four components: The first one is the encoder which receives the data and converts it to (SDRs). The second one is the Spatial Pooler which receives the outputted SDRs from the encoder and learns the spatial features of the passed SDR. The third one is the Temporal Memory which learns the temporal changes in the SDRs by activating

[3]http://nupic.docs.numenta.org/1.0.0/guides/swarming/running.html

particular cells in each column. The last component of CLA is the CLA Classifier, SDR Classifier which decodes the state of the CLA and produces predictions. HTM models require training and testing data that properly labelled, hence a dataset that is representative of the application at hand is needed to be developed in order to test, analyse and validate the application of HTM models in the context of smart homes, chapter 4 discusses the generation of such datasets.

Chapter 4

SIMADL: Simulated Activities of Daily Living Dataset

4.1 Introduction

Recent developments in technology have increased the adoption of smart devices and sensors in smart homes. With the realisation of the Internet of Things paradigm (IoT), the number of these internet-connected devices is likely to grow. In a study conducted by (Gartner, 2017), the number of connected Things is 8.4 billion devices in 2017. This number grew by 31% from 2016 and the study predicts that the number will continue to grow and will reach 20.4 billion connected devices by 2020. Moreover, the spending on IoT services that provide design, development, and implementation of IoT solutions was estimated to reach $273 billion by the end of 2017.

With the widespread usage of smart devices in smart homes, these environments will generate an enormous amount of streaming data. These generated data has the potential to provide novel services to the smart home inhabitants to improve their standards of living. These services can benefit from the analysis of this generated data.

Machine learning has been widely applied to develop probabilistic and statistical methods and sequence-learning algorithms to classify and predict ADLs of inhabitants. Nowadays, machine learning models and their contribution to the Internet of Things (IoT) applications are becoming one of the most active and interesting research areas (Alshammari *et al.*, 2018a). The smart home is one of the most prominent applications of the IoT paradigm. There are many advantages for adopting smart home technologies such as monitoring energy consumption, security, automation, entertainment, eldercare, etc. To implement machine learning techniques in any of the previous applications, a

representative dataset for that application is required. The dataset will be used to train and test the machine learning models to evaluate and validate their performance.

There are real smart home datasets available in the literature (e.g., (Alemdar *et al.*, 2013b; CASAS, 2009; PlaceLab, 2005; Chavarriaga *et al.*, 2013)), however, they lack the flexibility to cope with the recent advancements in sensor techniques, and they are costly to build and construct. Up to the knowledge of the author, there is a lack of real-world datasets targeted at classification, in the context of smart homes that are publicly available (Cook *et al.*, 2009; Bouchard *et al.*, 2010; Synnott *et al.*, 2015).

Smart home simulation tools are an alternative to constructing real smart homes. These tools allow the researcher to design a smart home suitable to the application that they are investigating to generate a representative dataset. There is less cost and effort involved in this process, and the smart home simulator can cope with new emerging techniques. However, many of these simulation tools are not available in the domain as an open-source project, or they lack the flexibility and accessibility for both the researchers and the participants.

The simulation tools regarding dataset generation approache can be categorised into two approaches, model-based and interactive approaches. The model-based approaches generate datasets using pre-defined scripts that generate events, probability of the occurrence of events, and their duration. On the other hand, the interactive approaches capture the sensor activities and log them to the dataset in real-time. Examples of model-based approaches include (Lee *et al.*, 2015; Kormányos & Pataki, 2013; Bouchard *et al.*, 2010). Examples of interactive approaches include (Synnott *et al.*, 2014; Ariani *et al.*, 2013; Fu *et al.*, 2011).

The two approaches suffer from disadvantages for the researchers and the participants alike. Model-based approaches allow the researcher to generate big datasets in short periods of time. However, the generated datasets do not capture realistic and fine-grained interactions that happen in real smart homes. The interactive approaches can capture these fine-grained interactions because they capture the output of the sensors directly to the dataset. However, the interactive approaches produce smaller datasets and take more time for the participants to perform their habits. Most of the interactive tools focus on context-awareness applications and not on generating datasets.

OpenSHS is an open-source, 3D, cross-platform simulation tool that follows a hybrid approach and combines the advantages of both approaches. It allows the researcher to design a smart home model specific to their research problem and generate a sufficiently large dataset in reasonable time while retaining the fine-grained interactions that the participants are performing.

This chapter presents datasets generated by OpenSHS for classification problems. The remainder of this chapter is structured as follows: Section 4.2 presents the related work in the literature. Section 4.3 explains OpenSHS architecture and how it has been used to generate the datasets. Section 4.4 presents the methodology to generate the datasets. Section 4.5 provides a description of the generated datasets. Section 4.6 presents ADLs' classification . Section 4.6.1 presents ADLs' classification in OpenSHS. Section 4.7 provides a usability study.

4.2 Related Work

In this section, the available real datasets in the literature are reviewed as well as the simulation tools that allow the researchers to generate synthetic datasets.

4.2.1 Real Datasets

(Alemdar *et al.*, 2013b) published the Activity Recognition with Ambient Sensing (ARAS) dataset which is a real dataset for complex scenarios of multi-residents. The dataset was captured for the duration of two months for two different houses and each house had two inhabitants. ARAS dataset was used to assess ADLs classification algorithms (Alemdar *et al.*, 2013b).

The Centre for Advanced Studies in Adaptive Systems (CASAS) is a project for creating real smart homes for the researchers in this field. (Cook *et al.*, 2013) designed a simple and lightweight toolkit called "smart home in a box". The components of this toolkit are assembled in a single small box and easily installed in a home to be able to provide smart tasks. They have installed the toolkit in 32 smart homes and generated several datasets. The datasets are publicly available online (CASAS, 2009).

The TigerPlace (Skubic *et al.*, 2009b) conducted a study on the ageing population. They used passive sensor networks that were installed in 17 flats within eldercare facility. They used many kinds of sensors such as motion sensors, pressure sensors, etc. In some of the flats, the collection of data took more than two years.

Some datasets focus on wearable technologies to monitor and acquire the activities performed by the participants. The Smartphone-Based Human Activities Recognition (SBHAR) dataset (Anguita *et al.*, 2013) is one example of such datasets. The authors collected the accelerometer and gyroscope data of 30 participants who performed several ADLs using a smartphone. Casale *et al.* (2011) and (Bruno *et al.*, 2013) are other examples of similar datasets.

The Intelligent System Laboratory (ISL) (Van Kasteren *et al.*, 2010) generated a dataset from three smart homes in which a single participant was performing his ADLs. The dataset represent around two months' worth of data. The first smart home had 14 sensors, the second had 23 sensors, and the third had 21 sensors.

Using a camera feed to capture a participant activity is another approach to recognise ADLs. (Pirsiavash & Ramanan, 2012) presented a dataset of one million frames captured from a wearable camera that represent a first-person view. The data was gathered from 20 participants who performed unscripted ADLs in their homes.

The ContextAct@A4H dataset (Lago *et al.*, 2017) is an example of recent datasets that focus on ADLs. The dataset was generated using a real apartment equipped with many sensors of different types. The dataset consists of one week's worth of captured data during the summer season and three weeks of the fall season. The authors proposed a new annotation method using temporal logic.

(Chavarriaga *et al.*, 2013) offered the OPPORTUNITY Dataset for Human Activity Recognition, which is a real dataset derived from wearable, object, and ambient sensors. There are limited established benchmarking problems for activity recognition. They provided a common database to validate the performance of various machine-learning models. This dataset allows researchers to test and report the performance of their algorithms or new emerging methods. The data was read from movement by the sensors as the inhabitants performed regular daily activities. The data was read from different sensors, and the real activities were recorded. However, they had difficulty in capturing more interesting, accurate annotation and fine-grained details.

(Micucci *et al.*, 2017) presented a new dataset that focused on wearable technologies to observe and acquire human activity recognition and fall detection using an Android smartphone. The authors collected the accelerometer data of 30 participants. Almost all of them were female with a wide range of ages in the 18-60 age range. The dataset contained 11,771 samples of human activities and fall detection. The activities of daily living (ADLs) consisted 7,579 samples and 4,192 falls totalling 11,771. They have combined the data into nine different activities of ADLs and eight various labelled activities of falls. Every accelerometer entry has a certain label of ADL (e.g. walking, sitting, or standing) or the kind of fall (e.g. forward, syncope, or backward). The UniMiB SHAR dataset allows researchers to study many problems, for example robust features to deal with falls or the subjective case.

(Youngblood *et al.*, 2005b) presented MavPad and MavLab intelligent environments. MavPad and MavLab datasets were gathered from intelligent home environments. MavPad is a dataset generated from a full-time student inhabitant in a campus apartment.

For MavPad, to control devices in the home, they used 25 controllers, and there are many sensors used in the apartment, such as temperature, light, smoke detection sensors, etc. The MavLab environment includes meeting rooms, a bathroom, a lounge, etc. For MavLab, automation techniques and 54 X-10 controllers were used. Every context, for example light, temperature, and door/seat sensors, has a current state for the sensors, and it is specified by the sensor status.

(Alhafidh *et al.*, 2018) applied some machine-learning algorithms to predict an inhabitant's next event. To assess the proposed algorithms, they used their own dataset that is similar the MavPad dataset. The dataset was captured from sensors in an apartment. They used two kinds of sensors to predict an inhabitant's activity on a lighting actuator in a bathroom that had seven sensors installed in the apartment within the local area first and then 86 sensors located in whole apartment. The data collection took more than one month, and the dataset was split into training data for one month and test data for one week, where the models learn from the data. But their dataset not available in the public domain.

(Ordónez *et al.*, 2013) generated datasets that capture real-world activities from similar sensor devices in two homes. Each home has extremely diverse settings and layouts. They used sensors that were installed to observe the inhabitant. They selected the type of sensor based on simple installation and the least intervention. They focused on using the wireless sensor networks in smart home environments to scale equal things. They used passive infrared sensors to reveal movement in a certain place at home. Reed switch sensors involve actions such as opening and closing a door.

(Van Kasteren *et al.*, 2008) generated a dataset from one house. The dataset was captured for the duration of 28 days, and the house had one inhabitant. They used 14 state-change sensors that were installed in the apartment. They deployed many types of digital sensors, such as door sensors, cupboard sensors, etc. The number of sensor events was 2,120. In addition, they used activity examples. The authors proposed an annotation method by the occupant using a Bluetooth headset combined with speech recognition software. They categorised the data into seven activities: leaving the house, toileting, showering, sleeping, preparing breakfast, preparing dinner, and preparing a beverage. When there is no activity label annotated, the label is indicated as idle.

(Cook & Schmitter-Edgecombe, 2009) presented a smart apartment testbed on the WSU campus. The smart apartment has several rooms. The apartment has many digital and analogue sensors distributed over each room, such as temperature sensors, motion sensors, water-flow sensors, and observation of oven use sensors. The real data was gathered from 20 participants/undergraduate students who performed their ADLs. Each participant performs the following five activities: telephone use, hand washing, meal

preparation, eating and medication use, and cleaning. They used the sensor network during the experiment time, and the sensor outputs and state of the different devices were captured using the sensor network and stored in an SQL dataset.

(Quesada *et al.*, 2015) designed and developed datasets from ADLs in smart homes that were split in many sections (single, interleave, and multi-occupancy). The dataset was generated from the smart lab of the University of Ulster. They deployed a sensor network with 14 sensors distributed throughout a kitchen and a living room. The laboratory has two kinds of sensors: pressure and contact sensors. The sensors produce data, and the state of various devices are captured and stored in an SQL database. The dataset was captured from real activities for the duration of one week by multiple inhabitants. The dataset includes nine ADLs: drinking a glass of water, preparing tea with a kettle, preparing hot chocolate in the microwave, drinking a glass of milk, calling by telephone, preparing a hot snack in the microwave, preparing a cold snack, watching TV, and washing dishes. The dataset was manually labelled by the occupant to determine the move from one activity to another.

(Roggen *et al.*, 2010b) equipped a sensor network with 72 sensors distributed around several rooms and objects and on the body. The real data was gathered from 12 participants. Each participant performed a certain morning context in real time. The average time for performing ADLs per participant was around two hours, totalling 25 hours of sensor data. The number of record samples for interaction with objects and the environment was around 14,000.

(Singla *et al.*, 2009) presented a smart apartment testbed on the WSU campus. From different sensors, the dataset captured real ADLs that were performed by 20 participants in the smart apartment. It is an interleaved activity class dataset in which one inhabitant performed an activity while another inhabitant simultaneously performed another activity. They chose eight ADLs that exist in clinical questionnaires. The activities in the dataset are filling a medication dispenser, watching a DVD, watering plants, conversing on a telephone, writing on a birthday card, preparing a meal, sweeping and dusting, and selecting an outfit. This dataset can be used to evaluate researchers algorithms for health monitoring in a smart home domain.

(Singla *et al.*, 2010) developed a multi-resident (multi-occupancy) class dataset, in which two of the inhabitants are carrying out their activities concurrently. They used a smart apartment testbed on the WSU campus. The dataset was gathered from real activities of two co-occupants. It was performed by 40 participants. The dataset was manually labelled with the activity ID and the person ID of the occupant. The tester took care to time the recorded activities. They choose 15 ADLs that are always registered in clinical questionnaires.

The dataset is divided into two parts with activity name and type as follows: (Person A) filling medication dispenser (individual), moving furniture (cooperative), watering plants (individual), playing checkers (cooperative), preparing dinner (individual), reading a magazine (individual), gathering and packing picnic food (individual); (Person B) hanging up clothes (individual), moving furniture (cooperative), reading a magazine (individual), sweeping the floor (individual), playing checkers (cooperative), setting the table (individual), and paying bills (cooperative). Activities that allow both inhabitants to cooperate to perform specific activity are labelled cooperative, while other activities in the dataset can be carried out individually or in parallel.

4.2.2 Simulation Tools

(Synnott *et al.*, 2015) conducted a survey of existing simulation tools for generating datasets in a smart home environment. They found that, due to the sensors technology cost, availability limitations, time considerations and finding the optimal sensors configurations, simulation tools are valuable assets to have for smart home research. The authors also identified that most of the available simulation tools focus on context-awareness applications and not on generating representative datasets. Moreover, supporting multiple inhabitants was one of the features lacking in current simulation tools.

(Cook *et al.*, 2009) presented some challenges facing the evaluation of machine learning performance and pervasive computing techniques. The authors identified the need to have real datasets and there is a lack of real datasets in the literature.

(Bouchard *et al.*, 2010) designed a 3D smart home simulator for activity recognition to overcome the limitations of creating real datasets in a smart home. Many pre-recorded scenarios were captured from clinical experiments and used to generate datasets.

To evaluate activity recognition algorithms, researchers require good representative datasets. Due to the high cost of building real smart homes and due to privacy and ethical issues with the human subjects, (Helal *et al.*, 2011) developed an event driven simulation tool for researchers in the smart home domain. The developed simulator is called “Persim” and it can generate realistic datasets for complex scenarios of the occupant’s activities.

An improved version of Persim was developed by (Helal *et al.*, 2012) called PerSim 3D. This tool helps to generate realistic datasets from the inhabitants’ activities in a smart home scenario. The major improvement was adding 3D simulations of the inhabitant, sensors and actuators. In addition, the tool supports the researcher by a Graphical User Interface (GUI) to envisage the activities in 3D.

The intelligent environment simulation (IE Sim) was developed by (Synnott *et al.*, 2014) to generate synthetic datasets that capture ADL of smart home users. IE Sim provides the researcher with a 2D graphical interface of the floor plan to design a smart home. The researcher can add different types of sensors such as temperature, pressure sensors, etc. Then, using an avatar, the simulation can be carried out to capture ADLs. The output of the simulation dataset is in the homeML (McDonald *et al.*, 2013) file format.

(Lundström *et al.*, 2015) extended an IE simulation that combined a probabilistic model (Poisson distribution) and the avatar control to increase the realism of the recorded data by the IE simulation. Thus, this extended version of the IE simulation adopts a hybrid approach that merges the model-based approach to generate large datasets in a short period of time, and the interactive approach, which is able to capture fine-grained interactions. The IE simulation is a 2D tool that simulates activities at a constant rate. This could be an issue when 3D motion data are required for the evaluated algorithms, such as in anomaly detection algorithms. They used a fast-forwarding feature that enables a simulation when the activity state has longer periods of low activity, such as sleeping, where there is no need to carry out the whole activity in real time. They used an interactive approach based on an avatar that controled it via a participant. The interactive approach has the benefit of recording fine-grained details. The participant can manage the place and time in which the activities happen. The avatar can be interacted with in the virtual environment via virtual sensors and actuators. The disadvantage of the interactive approach is the extended time required to generate a sufficient dataset.

(Ariani *et al.*, 2013) developed a simulation tool that uses ambient sensors to record the interactions of the occupants. The tool uses a map editor that allows the researcher to design a floor plan for a smart home by drawing shapes on a 2D canvas. The researcher is able to generate datasets for complex and different scenarios via simulation. The simulation tool could simulate binary motion detectors and binary pressure sensors. Simulations could assist rapid development, which would facilitate the design of a future smart home. They used the A* pathfinding algorithm to simulate the path of the simulated participant in the environment. This simulator can support researchers to determine which sensors are able to react to diverse simulated inhabitant movements. It used the PIR sensor to detect motion that is sensitive to quick changes in the environment; thus, if there are any variable factors, they will affect their response, such as the temperature, volume of the object in motion, the speed of movement, etc. This simulation tool is used to generate data with multiple occupants. During the simulation time, the sensor outputs are sampled at 5Hz and are stored in the database.

(Buchmayr *et al.*, 2011) developed a simulation tool that is facilely extensible to add new kinds of sensors. The simulation tool was used to generate and visualise simulated

datasets that capture data from normal and abnormal ADLs of residents. It is used for classification problems. They used a 2D approach because the floor plans are available in 2D, and it is easy to add new types of sensors on the floor plan. The tool can simulate binary switch, contact switch, and temperature sensors or just emit a signal on activation, while binary motion detectors and binary pressure sensors periodically emit signals after activation. During the simulation time, the sensor outputs are stored in a file or can be sent via the Internet to a server.

(Poland *et al.*, 2009) offered a simulation tool that uses virtual sensor distributions to capture the interactions of the ADLs of independent living for the elderly. It is a 3D smart home simulator for dataset generation. They developed the simulation tool to overcome the drawbacks/challenges of sensor distribution in a smart home. This approach provides the possibility of customising the right sensor deployment according to older people's needs.

(Youngblood *et al.*, 2005b) developed a ResiSim 3D simulator tool that provides real-time context simulation within a virtual smart environment. It was used to generate simulated datasets that capture ADLs of inhabitants, and it can be used to build models for automation. Additionally, they used the collected data from the simulator for visualisation to represent the collected data to best provide comprehension and analysis. The researcher can simulate certain scenarios in smart spaces with several objects. However, every object requires a separated simulation. The researcher can add and remove sensors to the virtual home. The ResiSim tool can be divided into three categories: (i) logical proxy as a real object (e.g. bed or chair), (ii) simulation objects, and (iii) user interface objects, which enable the researcher to interact with the virtual environment. The interface can be connected through the Internet to the ResiSim server.

4.3 OpenSHS

Most of the available simulation tools, discussed in Section 4.2.2 follows two approaches to generate synthetic datasets, model-based and interactive approaches (Synnott *et al.*, 2015).

The model-based approach relies on already defined statistical models of activities to generate synthetic data. The statistical model determines the order of events, the probability of occurrence, and the duration of activities. The model-based approach makes it easy to generate large datasets in a short period of time. The disadvantage of this approach is the lack of capturing fine-grained interactions and/or unexpected accidents that are common in real activities.

The interactive approach, on the other hand, can capture more interesting interactions and fine-grained details. This approach uses a virtual avatar controlled by a researcher, a human participant. The avatar moves and interacts with the virtual environment equipped with virtual sensors and/or actuators. These interactions can be passive or active.

An example of active interactions is opening a door or turning the light on or off. Another example of passive interactions is having a pressure sensor installed on the floor that detects the movements of the avatar without the avatar explicitly activating the sensor. The disadvantage of the interactive approaches is how long it takes to generate enough data: because of the nature of the approach, the interactions must be captured in real-time.

4.3.1 OpenSHS Advantages

Most of the simulation tools in the literature are not open-source, except for (Bouchard *et al.*, 2010), which makes it harder for the researcher to acquire the software and modify it to the experiment's need. In addition, having a 3D simulation adds to the realism of the conducted experiment.

OpenSHS is an open-source smart home simulator that allows the participants to simulate their ADLs in a 3D virtual environment. OpenSHS is developed with open-source and cross-platform techniques that makes it easy for the researcher to modify the tool and extend it according to their needs (Alshammari *et al.*, 2017).

The approach that OpenSHS uses to generate datasets can be thought of as a hybrid approach of the model-based and interactive approaches. OpenSHS offers a replication mechanism of the recorded ADLs which allows for a quick and large dataset generation, similar to the model-based approaches. The replications have fine-grained details as the activities are captured in real-time, similar to the interactive approaches.

OpenSHS has the flexibility to add different activity labels that can be customised by the researcher and tailored to their needs. It also has a fast-forwarding feature which facilitates the simulation of long inactivity periods. In this research, OpenSHS is used to generate the datasets for classification of ADLs problems.

4.4 Methodology

In this section, the design of the smart home and the contexts to be performed by the participants is presented, followed by the aggregation and generation of the datasets.

4.4.1 Smart Home Design

A smart home was designed consisting of a bedroom, living room, bathroom, kitchen, and home office, as shown in Figure 4.1. Each room has several types of sensors.

FIGURE 4.1: The design of the smart home.

The smart home is equipped with twenty-nine binary sensors, as shown in Table 4.1. Each binary sensor has two states, on (1) and off (0). The sensors can be divided into two groups, passive and active. The passive sensors do not explicitly require the participant to interact with them. Instead, they react to the participant movements and position. An example of this type is the carpet sensor. The carpet sensor turns on when the participant walks over it.

The other type of sensors are the active sensors. This type requires explicit action from the participant to change their state, for example, when opening a door or when turning on the light.

The activities' labels included in this dataset are: *sleep, eat, personal, work, leisure, and other*.

The participant controls a 3D avatar in first-person view and navigates and performs his/her ADLs in the virtual smart home environment. Throughout the simulation period, OpenSHS will capture and record the state of all the smart devices and sensors every second. Some activities take a long time, such as staying at the office for studying. OpenSHS provides a solution for this problem by implementing a fast-forwarding mechanism which enables the participants to quickly perform the long constant activities.

During the simulation, when a participant wants to change his/her activity, they can do that by using the dialogue shown in Figure 4.2. It is worth noting that, when the participants change their activity label, it does not immediately apply the change in the dataset. The activity label changes when one of the sensor's state has changed. This approach ensures a clean separation when the participant transits from one activity to another.

OpenSHS uses the concept of a context which is a specific time-frame of interest to the researcher to be simulated (Alshammari *et al.*, 2017). In this work has chosen to simulate the interactions of the participants in different contexts. On the weekdays, there are two contexts, one in the morning and the other in the evening. On the weekends, there are the same contexts during the day. Thus, there are four different contexts per participant. The day contexts are "morning" and "evening" contexts. The week contexts are "weekday" and "weekend".

TABLE 4.1: OpenSHS sensors.

#	Name	Type	Description	Active/Passive
1	bathroomCarp	binary	Bathroom carpet sensor	Passive
2	bathroomDoor	binary	Bathroom door sensor	Active
3	bathroomDoorLock	binary	Bathroom door lock sensor	Active
4	bathroomLight	binary	Bathroom ceiling light	Active
5	bed	binary	Bed contact sensor	Passive
6	bedTableLamp	binary	Bedroom table lamp	Active
7	bedroomCarp	binary	Bedroom carpet sensor	Passive
8	bedroomDoor	binary	Bedroom door sensor	Active
9	bedroomDoorLock	binary	Bedroom door lock sensor	Active
10	bedroomLight	binary	Bedroom ceiling light	Active
11	couch	binary	Living room couch	Passive
12	fridge	binary	Kitchen fridge	Active
13	hallwayLight	binary	Hallway ceiling light	Active
14	kitchenCarp	binary	Kitchen carpet sensor	Passive
15	kitchenDoor	binary	Kitchen door sensor	Active
16	kitchenDoorLock	binary	Kitchen door lock sensor	Active
17	kitchenLight	binary	Kitchen ceiling light	Active
18	livingCarp	binary	Living room carpet sensor	Passive
19	livingLight	binary	Living room ceiling light	Active
20	mainDoor	binary	Main door sensor	Active
21	mainDoorLock	binary	Main door lock sensor	Active
22	office	binary	Office room desk sensor	Passive
23	officeCarp	binary	Office room carpet sensor	Passive
24	officeDoor	binary	Office door sensor	Active
25	officeDoorLock	binary	Office door lock sensor	Active
26	officeLight	binary	Office ceiling light	Active
27	oven	binary	Kitchen oven sensor	Active
28	tv	binary	Living room TV sensor	Active
29	wardrobe	binary	Bedroom wardrobe sensor	Active
30	Activity	String	The current participant activity	
31	timestamp	String	The timestamp every second	

FIGURE 4.2: The activities selection dialogue.

4.4.2 The Participants

The participants in this work were chosen randomly. They also have experience with first-person games which will facilitate the learning curve of the tool.

The number of participants was 7, and the average time it took to conduct the simulation was 50 minutes ($min_{time} = 30min, max_{time} = 75min, std_{time} = 14.43min$).

For each participant, the following procedures were followed:

1. The researcher guides the participant and shows him/her the virtual smart home.
2. The participant is asked to play with the virtual smart home to get familiar with it.
3. The participant's familiarity with the virtual smart home is tested by asking him/her to perform specific tasks.
4. The actual simulation takes place, and the participant is asked to give his/her actual starting times for each context.
5. The participant is asked to complete the usability questionnaire.

4.4.3 Dataset Aggregation

To accelerate the process of generating the dataset, the participants are asked to perform several simulations of the same context. Since the activities of the participants are recorded in real-time, every simulation will be different and will contain unique information.

OpenSHS provides an aggregation algorithm that uses all the real-time recorded simulations to generate a new and random dataset but in a controlled manner (Alshammari *et al.*, 2017).

For each participant, this research has generated six datasets with unique parameters. The parameters used to generate each dataset are as follows:

1. Days: 30 and 60 days were chosen.
2. Start-date: 1 February 2016 were chosen.
3. Time-margin: the values 0, 5, and 10 were chosen.

The above parameters generated one month and two months worth of data. For the one-month set, three variants with 0, 5, and 10 time-margins were considered. The same goes for the two-month set. This ensures that the generated datasets are different in the time dimension. Table 4.2 shows a sample of the final dataset.

TABLE 4.2: A sample of the final dataset output.

Timestamp	**Bed Table Lamp**	**Bed**	**Bath Light**	**Bath Door**	...	**Activity**
2016-04-01 08:00:00	0	1	0	0	...	sleep
2016-04-01 08:00:01	0	1	0	0	...	sleep
2016-04-01 08:00:02	0	1	0	0	...	sleep
2016-04-01 08:00:03	0	1	0	0	...	sleep
2016-04-01 08:00:04	1	1	0	0	...	sleep
2016-04-01 08:00:05	1	0	0	0	...	sleep
2016-04-01 08:00:06	1	0	0	1	...	personal
2016-04-01 08:00:07	1	0	0	1	...	personal
2016-04-01 08:00:08	1	0	1	1	...	personal
2016-04-01 08:00:09	1	0	1	1	...	personal
2016-04-01 08:00:10	1	0	1	1	...	personal
⋮	⋮	⋮	⋮	⋮	⋮	

4.5 SIMADL Description

In this Section the generated SIMADL datasets from the OpenSHS simulation tool for classification of ADLs problems in smart home is presented (Alshammari *et al.*, 2018b). Seven participants were asked to perform their ADLs. The datasets represent around one

month worth of data. The dataset consists of forty-two files. The naming convention used for the datasets files is d{x}-{y}m-{z}tm where:

- x is an index number to uniquely identify a dataset;
- y is the number of months generated; and
- z is the time-margin value.

Each classification dataset has a target column of the previously mentioned labels of the activities. In addition to the twenty-nine binary sensor readings, datasets have a timestamp column.

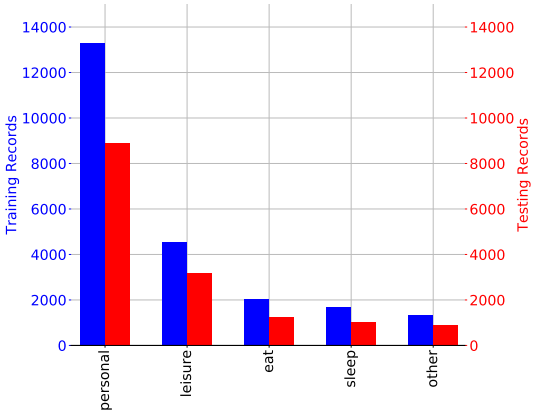
Table 4.3 shows a listing of the number of records for all the datasets generated excluding the header record. It is worth noting that, for each file in the classification dataset, OpenSHS generated the final output randomly from the record samples.

Figure 4.3 shows seven bar charts of the classification files. Each bar chart shows the proportions of the training records (the first 60%) and the testing records (the last 40%). Some files do not have all the labels included because the participants did not perform that activity, for instance, as shown in the dataset d1_2m_0tm where the participant did not perform the "work" activity.

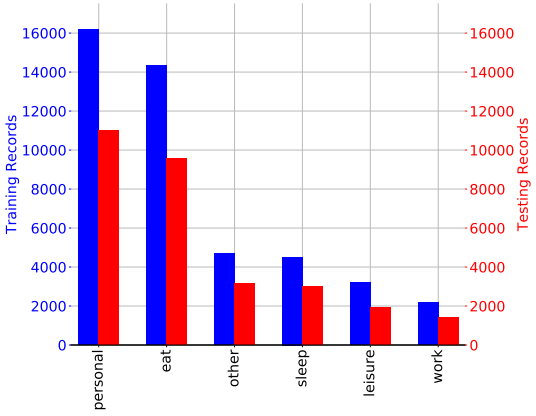
Figures 4.4, 4.5 and (Appendix A Dataset) show the frequency of the active sensor readings that are associated with the "leisure" label in the training and testing samples which shows that there are slight differences between the two. The remaining labels, figures, and dataset files are available online at `http://datasets.openshs.org` .

TABLE 4.3: The number of records for the forty-two files for classification datasets.

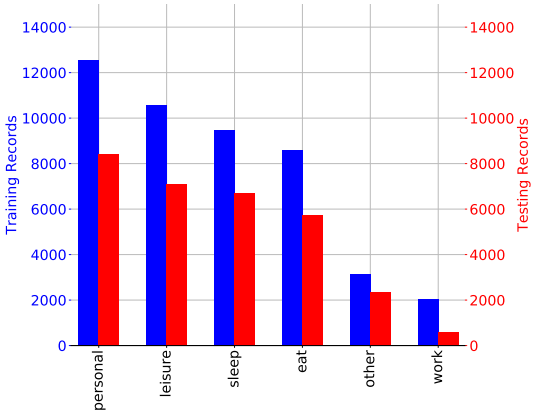
Name	**Classification Dataset**
d1-1m-0tm	18,800
d1-1m-5tm	18,966
d1-1m-10tm	18,828
d1-2m-0tm	38,204
d1-2m-5tm	37,532
d1-2m-10tm	38,012
d2-1m-0tm	37,332
d2-1m-5tm	36,261
d2-1m-10tm	35,687
d2-2m-0tm	75,183
d2-2m-5tm	72,302
d2-2m-10tm	73,526
d3-1m-0tm	39,832
d3-1m-5tm	42,526
d3-1m-10tm	40,730
d3-2m-0tm	77,328
d3-2m-5tm	83,346
d3-2m-10tm	79,933
d4-1m-0tm	40,232
d4-1m-5tm	40,015
d4-1m-10tm	38,629
d4-2m-0tm	80,033
d4-2m-5tm	79,171
d4-2m-10tm	79,176
d5-1m-0tm	27,762
d5-1m-5tm	28,008
d5-1m-10tm	28,450
d5-2m-0tm	55,577
d5-2m-5tm	56,200
d5-2m-10tm	56,919
d6-1m-0tm	81,859
d6-1m-5tm	85,763
d6-1m-10tm	84,672
d6-2m-0tm	165,596
d6-2m-5tm	165,038
d6-2m-10tm	167,282
d7-1m-0tm	49,282
d7-1m-5tm	49,605
d7-1m-10tm	49,769
d7-2m-0tm	100,544
d7-2m-5tm	100,498
d7-2m-10tm	100,502
Total	2,674,910



(A) d1_2m_0tm dataset

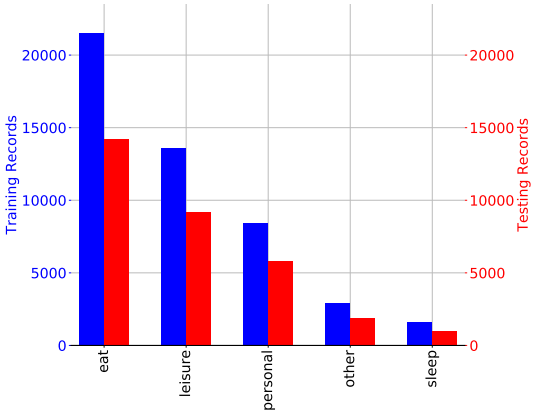


(B) d2_2m_0tm dataset

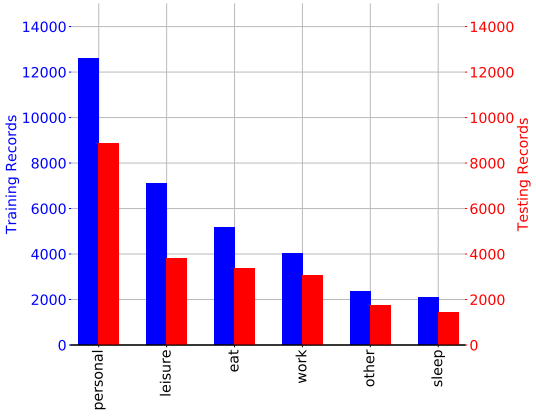


(C) d3_2m_0tm dataset

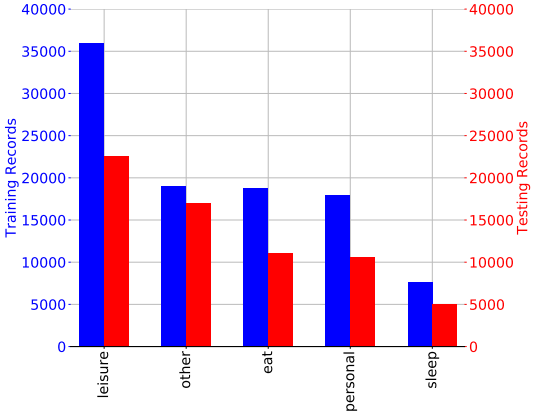
FIGURE 4.3: Analysis of the classification datasets with a 60%/40% split for training and testing records (d1-2m-0tm To d3-2m-0tm).



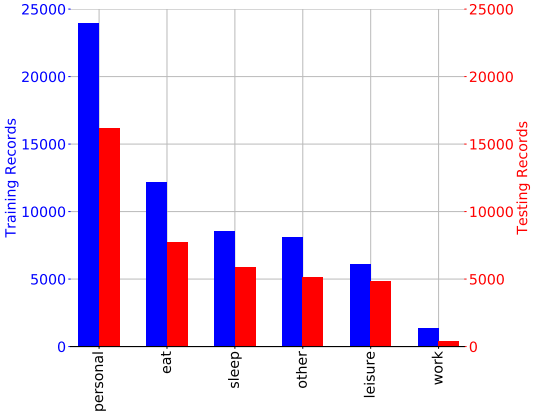
(D) d4_2m_0tm dataset



(E) d5_2m_0tm dataset



(F) d6_2m_0tm dataset



(G) d7_2m_0tm dataset

FIGURE 4.3: Analysis of the classification datasets with a 60%/40% split for training and testing records (d4-2m-0tm To d7-2m-0tm).

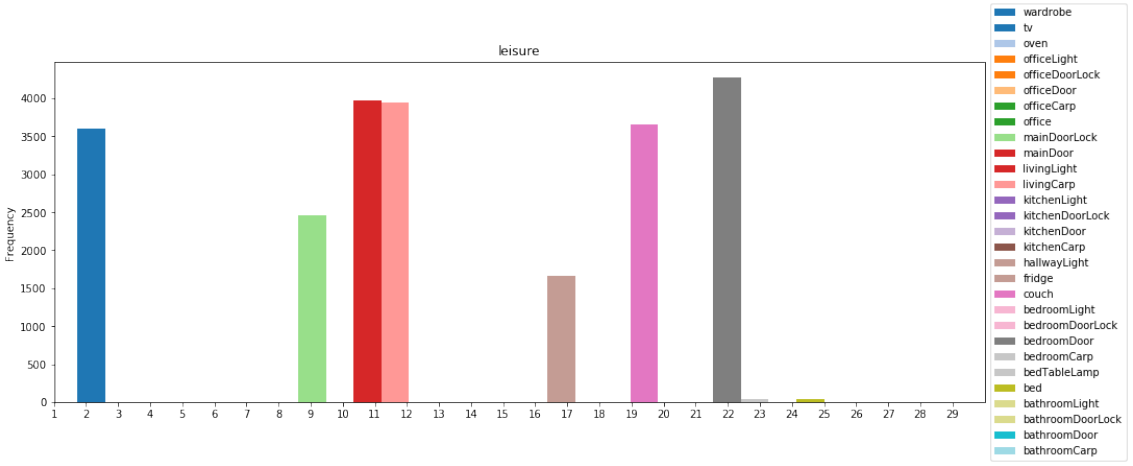


FIGURE 4.4: The sensor readings for the leisure activity in the training sample.

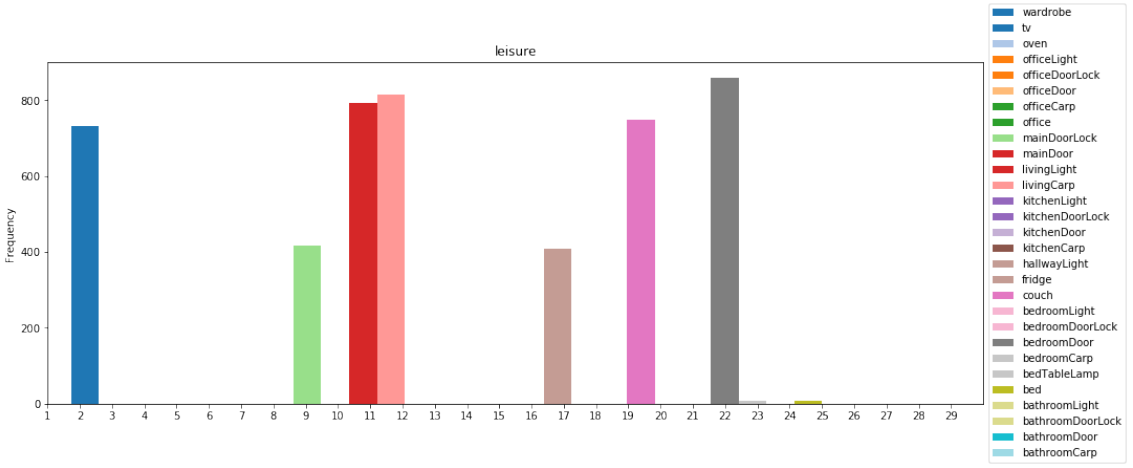


FIGURE 4.5: The sensor readings for the leisure activity in the testing sample.

4.6 Classification ADLs

Activities of daily living are tasks that the inhabitants perform in their daily routines. (Ni *et al.*, 2015) defined ADLs as "the things we normally do in daily living including any daily activity we perform for self-care such as feeding ourselves, bathing, dressing, grooming work, homemaking and leisure". This definition is focused on using smart homes, especially among older people and healthcare. To identify the limit of an activity and determine the type of physical activity performed by the occupant, which is an important feature for smart home applications, this requires a suitable description of the activities. (Ni *et al.*, 2015) proposed further details of a conceptualisation of activities. The first one describes the classification of activity, while the second one describes activities that are relevant to their conceptualisation. The conceptualisation process labels activities by description.

1. Classification of activities: The ADLs in the smart home can be classified from events to analyse and determine to user behaviour at various patterns in smart environments. An event is an activity carried out by a resident or participant in a short period of time, such as opening the door, watching TV, etc. In the context of an inhabitant‘s behaviour in their smart home environment, the definition of an ADL behaviour can be complicated. The ADL behaviour changes from one inhabitant to another. Moreover, coarse-grained activity can be classified into many fine-grained activities, such as preparing meals. When there are children or older people, this activity will be divided into two or more fine-grained activities. For example, for the child activity, it could be “preparing meals with milk”, and for older people, the activity could be “preparing meals without sugar or salt”.

2. Composition of activities: The composition captures these complex ADLs with many variants of activities to form a composite activity. The sorting of simple activities might depend on the inhabitant‘s sorting habits. Each activity has a starting/end time, which can communicate with other activities to compose an activity (Meditskos *et al.*, 2016).

(Ni *et al.*, 2015), (Quesada *et al.*, 2015) identified events that are divided into the three different categories of activities: 1) a sequential (single) activity performed before or after another activity with time intervals between two activities, such as preparing coffee and watching TV, 2) interleaved activities in which the inhabitant performs the activity while doing another activity, such as opening the door while watching TV (these interleaved activities are very different from the same activity), and 3) synchronous (concurrent) activities are carried out by multiple inhabitants at the same time, for example multiple residents can study in an office simultaneously.

4.6.1 Classification of ADLs in SIMADL Dataset

Each participant performs a simulation of his/her ADL classification to represent ADLs in various contexts by the OpenSHS interface module. OpenSHS allows the participants to use a context in which an inhabitant’s daily routines in a certain time period can be simulated. There are many contexts, such as “morning weekday”, “morning-weekend”, “evening-weekday”, and evening weekend contexts. The timestamps and dates are default, but the participant is able to change the default setting if necessary. OpenSHS provides the participant with six activity labels: sleep, eat, personal, work, leisure, and other. Each participant classifies the appropriate activity for each specific task. Before the simulation was conducted, the participant can choose the activity button shown in Figure 4.2. Through the simulation interval, when the participant performs a specific

activity, OpenSHS will capture and record the state of all the smart devices and sensors in the simulation every second. Some activities need to take a long time, such as staying at the sofa to watch tv. These activities are more difficult to simulate in real time. OpenSHS provides a solution for this problem using a fast-forwarding mechanism that enables the participant to identify the time for this activity. The tool will repeat the state of all the smart devices and sensors based on the specified time period. The participant should be familiar with the choice of classification of activities. During simulation, when a participant wants to change from one activity to another, the participant will be able to perform that change using the dialogue shown in the figure. To ensure a clean transfer of one activity to another activity, OpenSHS will not directly apply a change in the dataset. The activity label waits until the sensor state is changed. The participants are performing several simulations of the same context. It records the activities of the participants in real time. When participants perform the simulation, each participant simulates his/her activities in different ways. The participants are able to aggregate their generated sample activities. OpenSHS provides an aggregation algorithm that uses all the real-time recorded simulations to generate huge datasets in a short time during simulation. Using an aggregation algorithm that integrates two approaches, a model-based approach and an interactive approach, this feature enables OpenSHS to combine the advantages of both approaches. OpenSHS allows the participant to increase or decrease the activity labels, when the participants want to add more ADL classifications, OpenSHS enables the participant to determine several activity labels in the virtual smart home environment, which is customised by the researcher, based on the experiment needs. Additionally, the tool can be expanded to cover new emerging technologies.

4.7 Usability Study

Assessing the usability of any system is a tricky task because it involves quantisation of qualitative constructs, such as the participants opinions regarding the ease of use of OpenSHS.

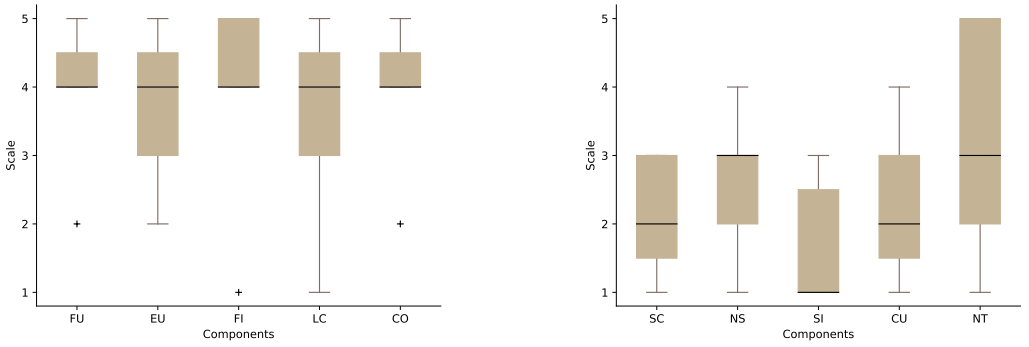
This research adapted the widely used System Usability Scale (SUS) Brooke *et al.* (1996) in order to assess OpenSHS usability from the stand point of the participants. There are ten statements that each participant will express their agreement/disagreement (similar to Likert scale). In order to calculate the SUS score, these statements are divided into two groups. The odd numbered statements (statements number 1,3,5,7,9) and the even numbered statements (statements number 2,4,6,8,10). The odd statements are positive and if the participant agrees with them, this will increase the overall score of the system.

On the other hand, the even statements are negative and will decrease the overall score if the participant agrees with them.

1. **Frequent use (FU)**: I think that I would like to use this system frequently.
2. **System complexity (SC)**: I found the system unnecessarily complex.
3. **Ease of use (EU)**: I thought the system was easy to use.
4. **Need for support (NS)**: I think that I would need the support of a technical person to be able to use this system.
5. **System's functions integration (FI)**: I found the various functions in this system were well integrated.
6. **System inconsistencies (SI)**: I thought there was too much inconsistency in this system.
7. **Learning curve (LC)**: I would imagine that most people would learn to use this system very quickly.
8. **How cumbersome the system is (CU)**: I found the system very cumbersome to use.
9. **Confidence in the system (CO)**: I felt very confident using the system.
10. **Need for training before use (NT)**: I needed to learn a lot of things before I could get going with this system.

Our sample consists of seven participants who were asked to answer the SUS questionnaire. Six were male, and one female and average age of the participants was 30 ($min_{age} = 21, max_{age} = 35$). 100% of the sample did play first-person games and all the participants reported that they use their computers on daily basis. Figure 4.6 shows the results. The SUS score for this group is 68.21 out of 100 ($score_{min} = 55, score_{max} = 80$).

As OpenSHS is based on a game simulation tool, the rationale to choose participants is based on finding candidates who can use computing game simulators without requiring excessive time for training. The limitations are that the average age of the participants was 30 and to repeat the same for a different scenario e.g. with elderly people, the experiments would need to be repeated using a different set of participants.



(A) The positive components. (B) The negative components.

FIGURE 4.6: The result of System Usability Scale (SUS) questionnaire.

This research has chosen SUS because it is easy measure to manage the participants, it is suitable to use on few sample volume that provides reliable scores and it is robust. Moreover, it links to other measures of usability e.g. the Software Usability Measurement Inventory (SUMI) Brooke *et al.* (1996). The questionnaire contains 10 questions, which are divided into positive and negative questions to prompt the participants to read the questions more carefully. The usability study results are promising but revealed that there is a room for improvements. The learning curve (LC) component of the questionnaire was the lowest scoring component and therefore needs improvement. The results also show that the participants needed some support from a technical person to be able to use the system as indicated by the (NS) component. Some participants may need extra training before using this tool.

4.8 Conclusions

This research introduces datasets for the smart home research community, it is for classification. The datasets are generated using a simulation tool (OpenSHS), and seven participants simulated their ADLs. The collection of the generated date accumulates to 31 days worth of patterns for datasets. Representative smart home datasets, such as the ones presented in this research, have direct machine learning applications, mainly for the training, testing and validation of new models. Different datasets are needed depending on the machine learning target application, i.e., classification, clustering, prediction or anomaly detection. The contributed datasets can be used to validate machine learning models that perform classification tasks in the smart home domain. Classification tasks is applicable to many use cases such as automation, eldercare, healthcare,

entertainment, security, etc. As this data has been discussed it is important that it is encoded typically by hashing. Chapter 5 explores different mechanisms to address this.

Chapter 5

Proposed Multi-Region CLA Techniques

5.1 Challenges

This research proposes the application of CLA to classify and predict ADLs of inhabitants in a smart home environment. However, there are several challenges that have been addressed by this research, some of these challenges are related to the application domain i.e. smart home, and other challenges concern the effective application of the CLA in the context of a smart home. The data generated from different sensors in a smart home environment, represent spatio-temporal patterns indicating the inhabitants' activities. This research assumes binary outputs from these sensors. This stream of binary data flows continuously and might be prone to noise. The first of the challenges is to find meaningful spatio-temporal relations of such high-dimensional binary data. There is a need for a custom encoder that is more suitable for the nature of the data generated from smart home environments. Homes have changing layouts, and inhabitants perform their activities in various ways. The same activity might use a completely different sensor activation depending on the inhabitant's changing habits. To deal with this variability, there is a need to use supervised learning techniques that can cope with each individual smart home environments. The second key challenge is the selection of appropriate methods that can learn a sequence of binary data generated from multiple sensors to infer and predict different ADL patterns. The choice of the used classifier is crucial to produce meaningful predictions. The third key challenge is that there is a need for a better memory management of the CLA model to learn short and long-term patterns. This Chapter proposes two novel multi-region CLA techniques, featured by multiple spatial pooler and temporal memory regions that incorporate a hash encoder and an MLP

classifier. The first proposed technique includes two parallel spatial pooler and temporal memory regions. The second proposed technique includes one spatial pooler and three cascaded temporal memory regions. These settings are proposed to improve the long and short-term prediction performance in the context of a smart home.

To tackle the first identified challenge, to cope with high-dimensional binary data, the CLA was tested using standard NuPIC encoders on a smart home dataset. The results discovered the weaknesses of the standard NuPIC encoders with multi-columnar datasets. Standard NuPIC encoders used do not able to deal with binary data generated from datasets in a smart home. It is an issue due to the nature of the data in a smart home that has many sensors, hence there is a need to develop an appropriate encoder that can deal with these sensors data.

Other research (Alshammari, 2018) was conducted to develop a hash encoder to cope with this issue for anomaly detection application in a smart home. The CLA performance was improved for anomaly detection ADL in a smart home that showed increased by 47.5% compared to the standard CLA encoders. Although, in this research, the results of using the hash encoder with the CLA model, instead of the standard NuPIC encoders seemed promising and improved the performance of the model. However, based on the experiments presented in Chapter 6, this improvement was not substantial enough to supersede the performance of the state-of-the-art algorithms for classification ADLs in smart home.

Second Challenge: the choice of suitable classifier for smart home environment is crucial. Recently, state-of-the-art algorithms have produced better results than the SDR classifier used in the CLA based on the experiments presented in Chapter 6. In Chapter 6, different classifiers have been evaluated to select one of them. Our results show that the MLP classifier outperformed the CLA and SDR classifiers in a smart home environment. Thus, this research applied the MLP classifier that allows better prediction of the inhabitant's activities in a smart home environment.

Third Challenge: To accommodate for long and short-term patterns multi-region CLA techniques were proposed to learn input patterns from streaming datasets. In the first proposed technique, Each region learns the patterns to allow different memory models. In a parallel stream, and every region performs spatio-temporal grouping of the input patterns. The predictions of the two regions are grouped together with a classifier. The second proposed technique uses three regions. The first region learns the input and performs spatial and temporal of the input patterns, and the sequence of outputs (the active cells) are fed to the upper region TM to learn more abstract patterns and then passes its active cells to TM in a third region as well. The set of active cells in all TM concatenates into one array and passes it to the classifier.

5.2 Hash SDR Encoder

5.2.1 Introduction

Encoding plays an important role in HTM systems and affects its performance. The encoding region converts input data to SDRs because HTM systems demand data input in this form to enable them to learn and recognise patterns.

5.2.2 Flexible Encoder

(Purdy, 2016) showed that the biological sensory system possesses constant extent of values, which can be encoded. One such example is the cochlea. In humans, the ears have a domain of between 20 Hz to 20 kHz that can deal with sound. When sounds around us exceed the frequencies in this range, the ears become unable to handle it. One can create a new encoder and can determine the range, but there is no flexibility to change the range after the system learns this range. To cope with this issue, an encoder could be created that possesses a constant number of bits and the ability to encode without restriction regarding the range of values. This is done through the use of a hash function. The SDRs produced are quite different, although sometimes there is a small overlap between two SDRs that are far apart, but the HTM system can recognise these SDRs. The hash function maps input of any length to a fixed size output. The input is passed to the hash function, producing output called hash values or digests. Hash functions are used to identify different inputs and are beneficial for cryptography.

There are many non-cryptographical hashing functions, such as the xxHash CRC32 2 and the Adler-32. The xxHash function is faster and achieves the required conditions, such as estimating the collision and randomness of the produced hash values or digests. Thus, this research used the xxHash function (Collet, 2015).

5.2.3 Problem Definition for Hash Encoder

This research has tested the CLA using Algorithms API through the standard NuPIC encoders on a smart home dataset. The datasets comprised of many columns representing the sensors readings and the activity labels. The accuracy and the performance were evaluated using the standard NuPIC encoders. The results revealed weakness of the standard NuPIC encoders with multi-columnar datasets. Due to the nature of data in a smart home with many sensors, there is a need to develop an encoder to deal with these sensors. This study aimed to understand all of the aspects above in order to develop an

efficient encoder. A hash encoder was used to rectify the CLA shortcomings with multi-columnar datasets from a smart home and convert each record uniquely SDRs. The rules for good encoders mentioned previously were taken into consideration. The Hash encoder uses a xxhash function to keep deterministic property. Moreover, the sparsity and the dimensionality of the output is preserved. Thus, the last three required properties for good encoders are met. For every sensor, its value was encoded and their output was concatenated. The concatenation of the sensors' readings constitute the first half of the SDR. To achieve the first property, the semantic meaning, the standard NuPIC DateEncoder was used to encode the timestamp field. Then, the first hashed part of the SDR was concatenated with this date encoder output. The result is two parts SDR, where the first part encodes the sensors' readings and the second part is dedicated to the date field.

5.2.4 How Does the Hash Function Work?

Developing a new encoder requires encoding each record in the dataset that has a unique SDR while maintaining its deterministic property, fixed-in dimensions and sparsity. This research uses a xxhash function by feeding the input record to a hash function and converting it to a digest. The output digest is used to present w-bits in the resulting SDR. The figure 5.1 below shows how the hash works. First phase: each record in the dataset such as all sensors_status (e.g., [0,0,1,0,1,000,1]) that passes to a hashing function will produce a hash digest. Second phase: The digest will be used to assign the active bits in the SDR. The digest has many digits through each digit this was presented by the index of the active bit.

Sensor1	Sensor2	Sensor3	Sensor4	Sensor5 – (n-1)	Activity
0	0	1	0	1	Work

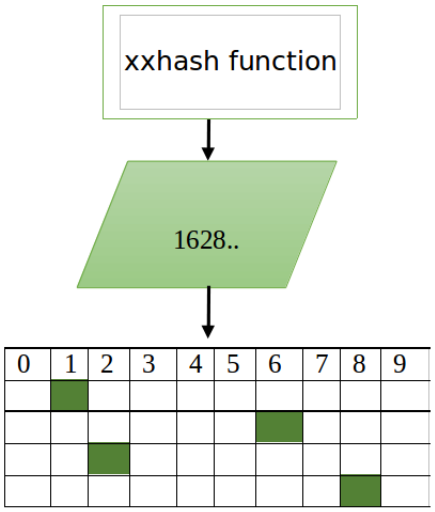
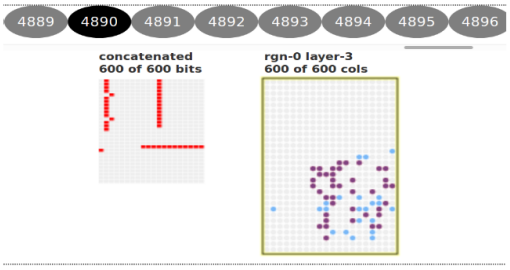


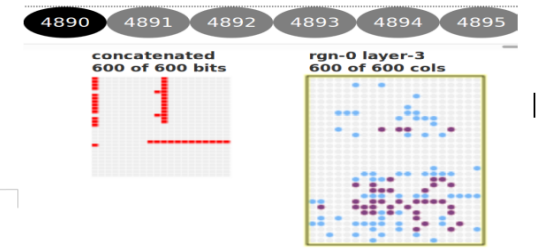
FIGURE 5.1: How the hash encoder works.

5.2.5 Analysis

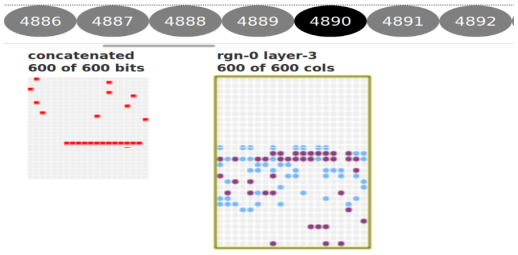
Figures 5.2 and 5.3 show a 2D sanity runs HTM models in the browser with interactive controls. It presents each bit in the SDRs generated through the encoder with different inputs.



(B) SDR Category Encoder1

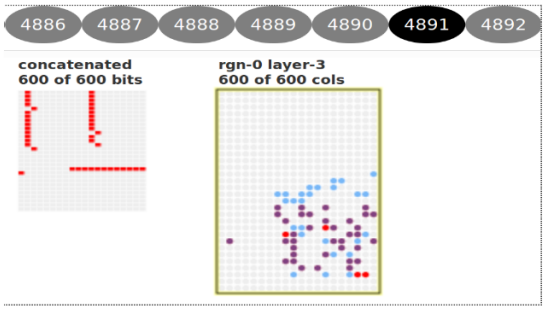


(C) Scalar Encoder1

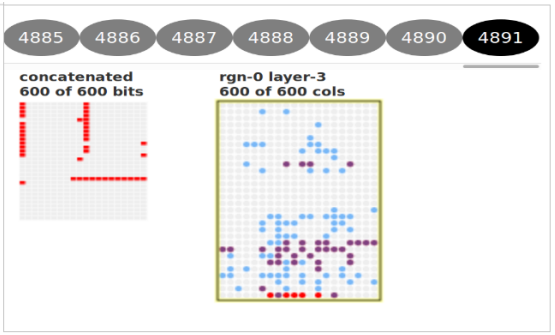


(D) Hash Encoder1

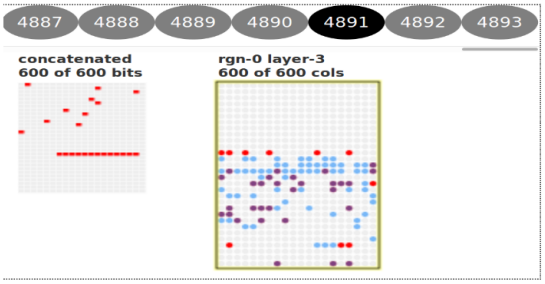
FIGURE 5.2: The output of SDR Category, Scalar and Hash Encoder for the same input .



(B) SDR Category Encoder2



(C) Scalar Encoder2



(D) Hash Encoder2

FIGURE 5.3: The output of SDR Category, Scalar and Hash Encoder for another input .

TABLE 5.1: Example of two different inputs.

Record		S1 - S8	S9	S10 - S21	S22	S23	S24	S25	S26	S27	S28	S29	Activity
A	→	0	1	0	0	1	0	0	0	0	0	0	sleep
B	→	0	1	0	1	0	0	0	0	0	1	1	personal

After reading around 5000 records from the dataset, two records were taken, as table 5.1 shows, in order to illustrate the difference between them. The record A represents sleep activity, and the record B represents personal activity. When looking to the above Figure 5.2 and Figure 5.3, they show the SDRs of the two records using ScalarEncoder and SDRCategoryEncoder. SDRs were generated for various activities such as sleep, personal and time, and each SDR was a little bit different. When the HTM system predicts daily activities over time, it faces obstacles to distinguish between activities. Figure 5.2 and figure 5.3 shows the SDRs using the hash encoder. The SDRs generated between them are extremely unique, which makes it easy for the HTM system to recognise these activities.

We note that the input space for the ScalarEncoder and SDRCategoryEncoder are similar and do not change drastically. However, the hash encoder has a completely different representation for each input. The parameters for the Hash encoder were determined, empirically through experiments, as follows:
N: dimensionality (total number of bits) is 600,
W: The number of active bits is 1, n: total number of bits for each sensor is 10,
As the smart home's model used in this research include 29 sensors and each sensor has 10 bits, thus totalling 290 bits for all the sensors. The remaining available bits for date encoder are 310 bits.

The hash encoder is able to work with high-dimensional binary data and convert each record uniquely into SDRs because the standard NuPIC encoders are not able to deal with high-dimensional binary data generated from datasets in a smart home. Hence, the results using the hash encoder were promising for anomaly detection of ADLs in the context of a smart home (Alshammari, 2018). It is anticipated that the uses of a hash encoder for classification of ADLs in the context of a smart home will improve the results.

The CLA with standard NuPIC encoders were tested on a smart home dataset. The results of the experiments, presented in Chapter 6, revealed weakness of the standard NuPIC encoders with multi-columnar datasets. This issue is due to the nature of data in a

smart home that has many sensors. There is a need to develop a new encoder capable of dealing with these high dimensional sensors. This research have used an existing Hash encoder and compared its performance with the models generated by NuPIC algorithm through standard NuPIC encoders. For more details about these findings and results on encoders, see the chapter 6.

5.3 Classifier

5.3.1 Introduction

In classification and predictions, applications are required to have a classifier to decode the state of the CLA model and generate significant predictions, where the classifier component is at the top of the CLA model. (Balasubramaniam *et al.*, 2015) illustrated a lack of existing classifiers. It is difficult to infer and predict an output from these SDRs, and there is excessive memory usage. The current implementation of the CLA, within the NuPIC framework includes several classifiers: the k-nearest neighbour (K-NN) classifier, the CLA classifier, and the SDR classifier.

5.3.2 Analysis

The KNN classifier and CLA classifier have been applied in many domains; however, to the knowledge of the authors, these classifiers do not appear to have improved the results of the CLA algorithm (Balasubramaniam *et al.*, 2015), (Yuwei, 2017). To analyse the performance with these classifiers, there are certain qualities for which the variances need to be investigated between these classifiers.

(Balasubramaniam *et al.*, 2015) illustrated the classifiers used in the NuPIC framework as the KNN classifier and CLA classifier. The KNN classifier maps the nearest neighbour and infers the output class, but there is an issue of storing each data point in memory, which is not suitable for large amounts of data; this implies the need to use excessive memory. The CLA classifier tries to map the active cells from the TM and learns from these active cells to predict future values. However, the results are still not efficient.

(Yuwei, 2017) at Numenta developed an SDR classifier to improve the prediction accuracy using a single feedforward neural network that uses the maximum likelihood estimation. It provides a full predicted distribution. It can be applied for classification and prediction, which produces better results than the CLA classifier and KNN classifier.

They clarified that the KNN classifier is used for categorical classification, and it preserves an SDR in memory but does not support the full predicted distribution. The CLA classifier output is the probability distribution of the predicted labels and votes on the label with the highest probability. However, the CLA classifier often predicts spurious outliers or anomalies.

From the previous studies, the major problem with NuPIC framework is the limitations in existing classifiers; which makes it difficult to infer and predict output from the produced SDRs. The KNN classifier uses too much memory; thus, it is not currently heavily used. The CLA classifier has spurious outliers in the prediction, while the latest SDR classifier tries to remove spiky outliers in the prediction, supporting the full predicted distribution.

The CLA and SDR classifier architectures are not equivalent to what happens biologically in the brain. They are not biologically inspired. The CLA and SDR classifiers have a small partial relation to biological use, and there are sequences of numbers to predict the number of steps into the future.

A (MLP) classifier was implemented to improve the CLA performance. The MLP is a traditional supervised learning algorithm that uses online decoding. It is also related to biological neural networks. The test results of the MLP classifier with the SDR classifier were compared in Experiment in Chapter 6 on the same evaluation methods used. In the following Section, the MLP classifier was presented that overcomes the problems that have been presented previously.

5.3.3 Multi-layer Perceptron

The (MLP) is a feedforward ANNs model, and it is a supervised learning model used for classification. The computational model of the MLP is inspired by the structure and functional sides of biological neural networks. The MLP maps the values of input data onto a value of suitable output. The data flow from the input layer to the output layer, and the MLP includes multiple hidden layers between the input and output layers. Every layer is fully connected to the next layer. Each node represents a neuron. Each input node has a weight. A weighted sum is calculated from all the inputs, and then the output is passed to a nonlinear activation function (Pedregosa *et al.*, 2011), (Haykin *et al.*, 2009), (Heaton, 2008).
After performing the activation function, the output from it is not the final value. It is passed to the next layer. Each input node contains connections, and each connection contains a weight. Many weighted interconnections exist between hidden layers, and the same input node has many connections that connect with many hidden layers with different weights. The MLP uses backpropagation for training the network to reduce the

error rate (Rumelhart *et al.*, 1986).
The MLP works in an online fashion for separation and classification between activities. However, because of the flexibility of ANNs and the ability to handle a nonlinear activation function, the neural networks represent an iterative method aimed to produce smaller errors during learning and training the neural network to reach weights that reduce the error rate. The units are connected between them in the form of interconnections. The MLP has some disadvantages, including the need for several hyperparameters to be set, such as the learning rate, number of iterations, and number of hidden layers.

- The ANNs structure:
  The ANN consists of an interconnected group of artificial neurons or processing units, which show similarity to the biological neurons in the human brain. The connections pass signals from one neuron to another, and each connection has a weight. Each neuron or unit receives a number of inputs and produces output, and the ANNs contains multiple hidden layers that are fully connected to the next layer. The multilayer network includes the input layer, hidden layer, output layer, and connections.

- Training or Learning in Neural Networks:
  There are two methods for training and learning in neural networks. The first is supervised learning, which provides the ground truth label with each example in the network. The network learns (adjusts weights) by knowing the difference between the actual output and target output. One of the most well-known learning algorithms is backpropagation, which is used for training the network. The second is unsupervised learning, where there are no labels available for the examples that are used to group similar examples into clusters.

5.3.4 Backpropagation Algorithm

In the ANN model, a set of inputs are provided for a set of desired outputs, starting with random values for weights (connection weights) and checking the results against the training data. The actual outputs and desired outputs are calculated and compared. When the results are not satisfactory i.e. the two outputs are not similar. Backpropagation calculates the error and determines the difference in the error in the network. It tries to adjust the connection weights and biases and goes backward over every step to reduce the error. The new weights are calculated using an equation based on the input value, learning rate, error rate, and initial weights. This process is reiterated numerous times. All parameters can be configured by the user, such as the epoch, batch size, learning rate, etc (Rumelhart *et al.*, 1986).

The values move to the output layer after the values have been calculated through the previous steps. A comparison is made between the values ($desiredvalue - actualvalue$) through the following error function equation 5.1:

$$E = target - output \tag{5.1}$$

Moreover, there is a need to correct and adjust the weights using the learning process from the network through the following equation 5.2:

$$Wi(Final) = Wi + \alpha\beta xi \tag{5.2}$$

where α is the learning rate and β denotes ($targetvalue - outputvalue$), Wi is weight and xi is input. It repeats these steps many times using feedforward and backward (feedforward back propagation), called epoch, until it reaches the best output values (Haykin *et al.*, 2009).

5.3.5 How ANNs Works

The model for a neural network is called perceptron. There are many inputs and weights, and this is basically a multiplication. $W1$ with $X1$ are multiplied, and $W2$ with $X2$, and so on. They are added together and then a bias component is added that is always weighted. If the sum is greater than zero, then the output will be One, but when the sum is Zero or less than Zero, then the output will be Zero.

5.3.6 Activation Functions

Activation functions process the input, weighted sum, and bias and calculate them. When neurons pass the threshold, these neurons fire in a layer. The activation function processes inputs into units or neural networks. The activation functions that have neural networks will be examined. There are five common activation functions for neural networks (Farhadi, 2017).

- Identity Function:
  The identity function states that f(x) = x, it is not an activation function, which is commonly used.
- Linear Function:
  The linear function (or step function) is suitable for binary classification tasks. The

terms 'positive' and 'negative' refer to the classifier prediction. All positive inputs make it One, and all negative inputs make it Zero, as in the following equation :

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 1 \end{cases}$$

- Sigmoid Function:
  There is a sigmoid that takes the form of a character (S). It is most commonly used in practical applications that have a range that is only positive numbers e.g. [0, 1]. Whatever the input is, it maps onto [0, 1]. This is used because it has nonlinearity, and it is easy to differentiate and derive. It is defined as follows equation 5.3:

  $$S(x) = \frac{1}{1 + e^{-x}} \tag{5.3}$$

  If one wants to represent positive numbers primarily in a neural network, the sigmoid function is an appropriate choice for the activation function.

- Hyperbolic Tangent Function:
  The hyperbolic tangent is over negative and positive numbers for the output of the function, which contains [-1, 1]. Therefore, if someone deals with values that are positive and negative, then this function is used. The derivative can be taken of both the sigmoid and hyperbolic tangent functions.

- Softmax Function:
  The softmax function is used for multiclass classification methods. It provides the probability distribution for various classes, and it divides each output such that the sum of the outputs is equal to one. This is used to impart probability. When there are have many outputs, it can pass outputs to this function and attain the probability distribution of each. This is useful for finding the most probable occurrence of the classification where the probability of a class is maximum.
  At every time step, the model provides an output, which is the probability distribution. It is summed to one at each time step. The label with high probability is the output.

5.3.7 Application of the MLP classifiers

The MLP classifier receives input that is a vector containing Ones and Zeros, an SDR from the TM as shown in figure 5.4. The input feeds into the network so that it arranges the sets of weights in a matrix layer by layer as shown below in figure 5.4. Every layer is fully connected to the next layer, and the input is multiplied by the weights. Then,

the activation function is applied, it uses a nonlinear activation function (the softmax function) to count the probability distributions of class labels. And the results pass to the next layer. The process continues in the same way until it reaches the last layer. The final output value is the prediction value or actual value. The output values are compared with the desired value (target) to determine the difference error among them. Then, the derivative is calculated backward through each layer and is adjusted using the weights to reduce the value of the error. This process is repeated until the error is smaller than certain threshold.

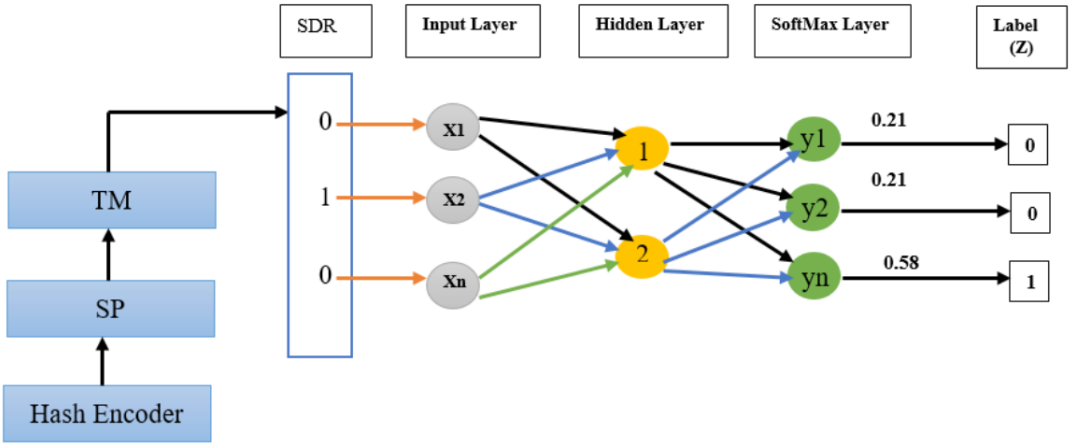


FIGURE 5.4: The CLA working with MLP classifier.

The NuPIC framework classifiers were presented, and the current implementation of the CLA has several classifiers: CLA, KNN, and SDR. A comprehensive review of the classifiers was conducted, and their benefits and drawbacks were discussed. In this section, this work tested the CLA using the NuPIC framework on a smart home dataset. The author proposed and applied the MLP classifier and compared its results with the standard ones. This research attempts to improve the performance of the classifier for CLA. The researcher proposed and applied the MLP classifier to provide a better prediction of the inhabitants activities in a smart home. For more details about these findings on classifiers, see the chapter 6.

5.4 Multi-Region CLA Techniques

5.4.1 Introduction

With the rapid growth in neuroscience over the last half century, the interest of scientists in neuroscience and cognitive psychology to understand how the memory biologically works has increased. Although there are many studies about memory, no overarching study has been conducted so far (Poo *et al.*, 2016).

This section proposes two novel multi-region CLA techniques, explaining biological short and long-term memory and presenting the capacity of sequential memory in the HTM model. It offers some previous studies for the CLA and the HTM theory in the literature. The following section introduces the purpose of developing the proposed technique and an analysis of the reasons behind developing it.

5.4.2 Biological Short and Long-term Memory

Our memories are stored in a short-term memory that is a part of the brain called the hippocampus. Then, those memories are consolidated in the neocortex for long-term storage. The study of the neural circuits conducted at Massachusetts Institute of Technology (MIT) discovered that memories are formed in the hippocampus, which conveys those memories to other parts of the brain in the neocortex. In a long-term memory, the cells become silent for a long time until they become functionally mature with time (Liu *et al.*, 2012).

In 1953,(Shah *et al.*, 2014) Henry Molaison underwent a delicate surgery in the brain in an attempt to treat epilepsy, but after the operation, he lost the ability to make new memories. He was able to remember memories only before the surgery. The doctors discovered that they had accidentally cut part of his hippocampus, but it gave neuroscientists the opportunity to infer and understand how memories are created, stored, and recalled. It was revealed that the hippocampus is very important for forming new long-term memories. The actual storage process for these memories is performed in a different part of the brain, in the neocortex. Scientists believe these short-term memories are formed and stored in the hippocampus before transfer to storage for long-term memories in the neocortex, vanishing from the hippocampus. In 2012, (Liu *et al.*, 2012) MIT researchers found a way to classify neurons called engram cells that include specific memories. This enabled the researchers to trace the circuits engaged in memory storage and recall. They have derived a technique that helps activate cells using light, and they

are able to turn cells on or off. They can excite the cells to remember a fear-conditioning situation (Poo *et al.*, 2016).

The researchers used this technique to excite the cells in a mice through a fear-conditioning case. The mice underwent minor electric shocks with the aim of creating a frightening event. Although light can be used to excite cells in mice to remember painful events in both areas, only the hippocampus area naturally activated the cells. After two weeks, the researchers tried to activate the cells in the hippocampus area using light, but the mice no longer exploited those cells to remember but activated the cells existing in the long-term memory in the neocortex.

5.4.3 Capacity of Different Forms of Sequence Memory

(Leibold & Kempter, 2006) illustrated the capacity of sequential memory. There is a limit to storing long sequences, and sequential memory can be changed in size to store minimal sequences. (Hawkins & Ahmad, 2016) explained in the HTM system that there is no concept to determine how many stored sequences there can be or why the capacity of sequential memory is not able to store long sequences. The HTM system is able to learn the changes or transitions between these inputs. Thus, the capacity of the HTM system can be determined by the number of transitions that can be used with mini-columns.

The capacity is calculated by cells per column/column sparsity. Each cell recognises a number of patterns through its basal dendrites. Where the basal synapses form lateral connections to nearby cells in the region, it learns the transitions of patterns over time. For example, when 2% of the columns are active, there are 10 cells per column, and each cell recognises 100 patterns on its basal dendrites. Thus, 50,000 transitions that can be stored ((10/0.02) * 100).

5.4.4 How Many Times Can the Same Input Represent Different Contexts?

To what extent can the number of the cells be increased in the TM? How can the number of cells per column affect the TM ability to recognise an input sequence? The SP activating a group of the mini-columns that have a number of cells. The TM employs these cells in mini-columns to make temporal relations between sequences of mini-columns generated by SP. If user has 40 mini-columns with three cells per column, a single input can be represented in (3^{40}) various contexts. The same word can be used in various sentences, for example (he eats an apple, and she eats a pear), the term eats is in both sentences. How can the CLA differentiate among them? The CLA uses one cell in the

same mini-columns. Such a word (eats) has individual representation in different contexts. Although the word is in the same mini-column, the cell changes to represent different contexts. The CLA has the ability to remember long sequences and recognise different contexts. Increasing the number of the cells in a mini-column provides the CLA with further capacity for sequence storage, and the CLA can retain long sequences in different contexts. Each cell in a mini-column has a specific job to perform with contextual meaning in the TM. (Hawkins *et al.*, 2016) illustrated that there is a possibility to increase and decrease the number of cells per column. When user increases the cells per column, it will increase the probability of encodings of input. They found, after empirical experimentation, that using more than 10 to 16 cells per column was enough for most cases; however, increasing cells in the mini-columns can be computationally expensive. Regarding decreasing the number of cells per column, when one cell per column is used, the CLA can remember one step back in history. There is limited capacity to recall longer sequences. Increasing the number of cells per column is suitable for recognising time-based sequences (temporal context), but using a single cell in the column is suitable for recognising spatial patterns.

5.4.5 Multi-Region HTM-Related work

The cortical learning algorithm (CLA) in the Numenta Platform for Intelligent Computer (NuPIC) originally used one CLA region; however, in this research, two regions for the hierarchy of the CLA were used. When using a single layer in the HTM system, there is a limit, where the CLA is only able to remember shorter sequences. To settle this problem, the HTM/CLA systems use multiple layers and more cells in a mini-column, allowing the CLA to have further capacity that enables the CLA to remember long sequences, like the columnar structure in a biological neural network (Jeff Hawkins, 2014).

(Agrawal & Franklin, 2014) tested the CLA with two regions, the feedback prediction used was from the higher region to set the cells in TM in lower region to predictive state. The performance of the CLA is better with two regions rather than one region.

(Kneller & Thornton, 2015) applied a HTM to detect anomalies and image reconstruction problems, they used the dataset of facial images. The authors proposed a new distal feedback method, it was used from higher region to provide more information about effectiveness of neighbouring regions at a lower level regarding the action of their lateral regions. They illustrated that the hierarchical message passing reinforces beneficial information, it cannot be implemented in one region. The feedback has accumulated after

the temporal pooler in higher level region has run. Their results suggest that the distal dendrite method seems good and it is further efficient than the predictive coding extension to the (CLA) PC-CLA approach.

(Rao & Ballard, 1999) presented The Hierarchical Predictive Coding (HPC) model that transfers message passing (feedback predictions) from the higher region into the lower region, higher level regions of a hierarchy predicts what input is coming next to send it to lower-level regions, when there are prediction differences with lower-level activity, it returns the prediction error. The errors can be used by the predictive estimator (PE) at each level to rectify its actual assessment that can generate the next input.

Neuroscience suggests that the brain is continuously generating prediction about future sensory inputs. (Friston, 2010) Friston's free-energy principle supports this, that the brain learns and tries to predict sensory input by the minimization of a free-energy bound on the probability of sensory inputs. When the system updates the probability for a hypothesis based on the statistics of its sensory input, the systems are doing Bayesian inference. Bayesian inference is suitable for the dynamic analysis of a sequence of data.

The CLAs attempt to construct generative probabilistic model that is able to predict future events or patterns based on the probability of the input it receives. Bayesian Inference aims to improve performance in the case of prediction uncertainty, thus using a top-down effect, as the areas or regions higher in the hierarchy have more abstract, hence passing the predictions to the lower regions improves their knowledge, the CLA white paper (Hawkins *et al.*, 2011) applied a one region system, one region does not meet the needs to apply Bayesian Inference. A two region CLAs can be applied using a Bayesian inference system, as the human brain biologically works (Agrawal & Franklin, 2014). (Hawkins & Blakeslee, 2004a) demonstrated that the lower regions of the hierarchy learn basic patterns, and these recognised patterns become more advanced and abstract in the higher regions.

5.4.6 Proposed Parallel Spatio-Temporal Memory Stream Technique (CLA2)

The proposed technique has two regions as shown in figure 5.5, called first and second region, both regions receive encoded input and is fed to a spatio-temporal grouping. The second region passes the active cells in the TM to be concatenated with the active cells in the TM in the first region. The original CLA, which is the implementation of the HTM theory, includes one spatial pooler and one temporal memory, the model can learn on one memory level by adjusting some parameters, such as the `CellsPerColumn`. The model learns either short-term or long-term patterns, not both of them. A novel

approach has been used in this research, which included two spatial poolers and two temporal memories to simulate short-term and long-term memories. The functions of the proposed technique is divided into the following steps:

- Encode input from the dataset.
- The spatial pooling (SP) in the first region learns from the output encoder.
- The TM in the first region learns, predicts, and feeds the indices of the active cells in the first region to the vector.
- Encode input from the dataset.
- The SP in the second region learns from the output encoder.
- The TM in the second region learns, predicts, and feeds the indices of the active cells in the second region to the vector.
- The vector concatenates both the active cells from TM in the first and second regions and then moves them to the classifier.

As shown in Listing 1, feeding the input record into a hash function to encode input and convert it to a digest builds the SDR. The spatial pooler in the first and second regions receives and learns from the SDR generated by an encoder. The temporal memory in the first region learns the temporal changes of the SDRs generated by the SP in first region and predicts and feeds the indices of the active cells in the first region to the vector. Temporal memory in the second region learns the temporal changes of the SDRs generated by the SP in the second region and predicts and feeds the indices of the active cells in the second region to the vector. The vector concatenates both the active cells from TM in the first and second regions and then moves them to the classifier. The listing 1 shows the pseudocode of the implementation of the proposed technique.

Algorithm 1 Parallel Spatio-Temporal Memory Stream .

$hash \leftarrow xxhash32Encoder(sensors)$
procedure CONSTRUCT-SDR($hash, n, w, p$)
    First Region
    $activeArray \leftarrow spatiaIPooler(sdr)$ ▷ Spatial Pooler learns SDR
    $activeColumns \leftarrow set(find_idxs(activeArray))$
    $TemporalMemory \leftarrow activeColumns$ ▷ TM learns the transitions of the SDRs
    $F \leftarrow GETACTIVECELLS$
    Second Region
    $activeArray \leftarrow spatiaIPooler(sdr)$
    $activeColumns \leftarrow set(find_idxs(activeArray))$
    $TemporalMemory \leftarrow activeColumns$
    $S \leftarrow GETACTIVECELLS$
    $Vector \leftarrow (F, S)$ ▷ Concatenates two arrays into one vector
    $Classifier \leftarrow Vector$ ▷ Moves the SDR in the vector to Classifier
end procedure

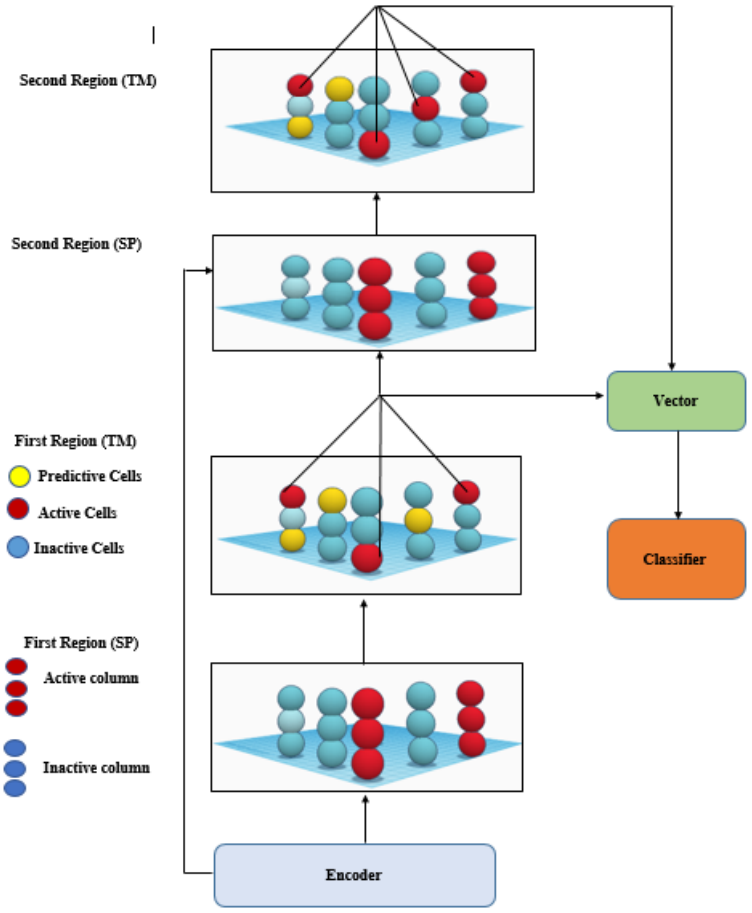


FIGURE 5.5: Parallel Spatio-Temporal Memory Stream.

5.4.6.1 Analysis

This research has been used the CLA in one region; however, state-of-the-art algorithms have recently produced better results than the CLA based on the experiments presented in Chapter 6. Thus, there is a need to use two regions in the hierarchy in the HTM model to improve performance. The first region learns short term patterns and second region learns long term patterns, in order to remove the limitation of memory management of the CLA. There are many model parameters for the TM in each region. After testing on all datasets, the performance of the CLA model improved when the value of `CellsPerColumn` decreased in the TM in the first region to learn short patterns and increased the cells per column in the TM in the second region to learn long patterns. It seemed that the performance of the model was affected by changing the number of cells. However, the performance of the CLA model has significant improvement. Figure 5.6 shows the performance of several (CLA2) with different values for the parameters `CellsPerColumn` in first region versus the `CellsPerColumn` parameter in second region.

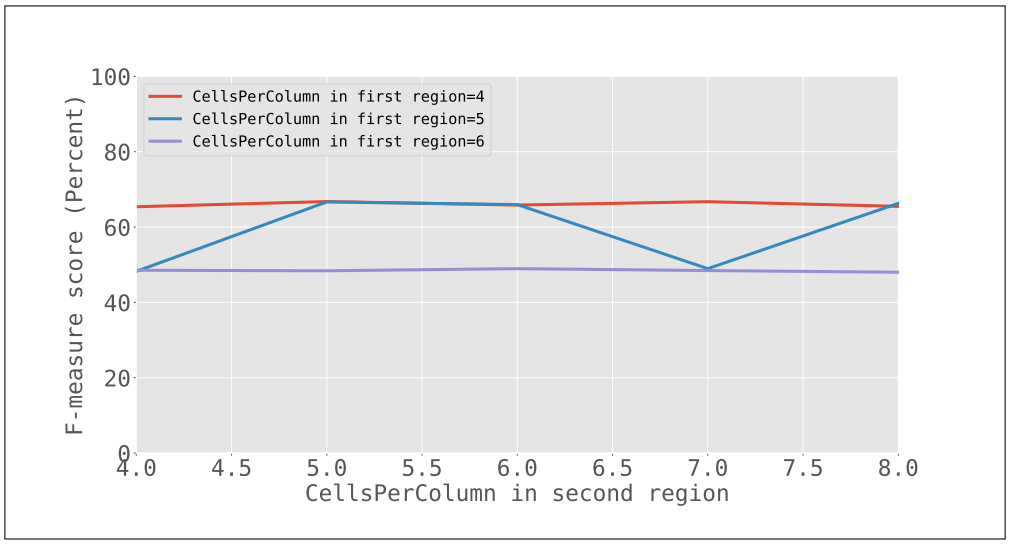


FIGURE 5.6: CLA2 results with different CellsPerColumn values with House-A.

Learning input patterns from streaming datasets with a one-region CLA is restricted by the number of cells per column applied which limits the model's memory. One-region is insufficient to manage short and long-term memory of such huge amounts of streaming data generated from smart home because the model learns either short-term or long-term patterns, not both of them. Tackling this issue requires memory management of the CLA model to learn short and long-term patterns. This research proposes a multi-region CLA technique to learn short and long-term patterns. When learning short-term patterns, the proposed technique uses a small number of `CellsPerColumn` to cope with short term events. Using such settings, the CLA can learn sequences and remember a short step back in history. Regarding learning long-term patterns, the proposed technique uses a large number of `CellsPerColumn` to enable the CLA to learn patterns and remember long- term sequences. Every region learns the patterns to provide various memory levels in a parallel spatio-temporal memory stream, and every region performs spatio-temporal grouping of the input patterns. The proposed technique concatenates the predictions of short-term patterns with predictions of long-term patterns into a vector and then transfers them to a classifier.

5.4.7 Proposed Cascaded Temporal Memories Stream Technique (CLA3)

The proposed technique has three regions as shown in figure 5.7. The first region learns smaller features and second, third regions are more abstract in order to learn and recognise patterns, the first region learns from the encoder's output, which is fed to spatio-temporal grouping, the first region passes its active cells to TM in a second region. The second region receives the active cells and learns it and then passes up its active cells to

TM in a third region. The third region receives the active cells and learns it. All regions pass the active cells in their TM to concatenate together in one vector. The reason for the message passing is to transfer feedforward predictions from the second and third regions to be concatenated with the first region in one vector that combines these predictions to improve knowledge. The functions of the proposed technique can be divided into the following steps:

- Encode input from the dataset.
- The spatial pooling (SP) in the first region learns from the output encoder.
- The TM in the first region learns, predicts, and feeds the indices of the active cells in the first region after they have been converted to column indices into the second region (TM).
- The TM in the second region learns, predicts, and feeds the indices of the active cells in the second region after they have been converted to column indices into the third region (TM).
- The TM in the third region learns, predicts.
- All regions feed the indices of the active cells in their TM to concatenate together in one vector.
- The vector concatenates all the active cells from TM in the first, second and third regions and then moves them to the classifier.

As shown in Listing 2, the input record is fed, encoded, and converted as noted above to build the SDR. Again, the spatial pooler in the first and second regions receives and learns from the SDR generated by an encoder. The temporal memory in first region learns the temporal changes of the SDRs generated by the SP in the first region and passes its active cells to the TM in a second region. The second region receives the active cells and learns the input and then passes it up to its active cells to the TM in the third region to learn it. All regions pass the active cells in their TM to concatenate together in one vector. The listing 2 shows the pseudocode of the implementation of the proposed.

Algorithm 2 Cascaded Temporal Memories Stream .

$hash \leftarrow xxhash32Encoder(sensors)$
procedure CONSTRUCT-SDR($hash, n, w, p$)
    First Region
    $activeArray \leftarrow spatiaIPooler(sdr)$ ▷ Spatial Pooler learns SDR
    $activeColumns \leftarrow set(find_idxs(activeArray))$
    $TemporalMemory \leftarrow activeColumns$ ▷ TM learns the transitions of the SDRs
    $F \leftarrow GETACTIVECELLS$
    Second Region
    $activeColumnsS \leftarrow list(get_sdr(10, F))$ ▷ Convert the active cells to column indices
    $TemporalMemory \leftarrow activeColumnsS$ ▷ The TM learn it
    $S \leftarrow GETACTIVECELLS$
    Third Region
    $activeColumnsT \leftarrow list(get_sdr(15, S))$
    $TemporalMemory \leftarrow activeColumnsT$
    $T \leftarrow GETACTIVECELLS$
    $Vector \leftarrow (F, S, T)$ ▷ Concatenates three the indices of the active cells into one vector
    $Classifier \leftarrow Vector$ ▷ Moves the SDR in the vector to Classifier
end procedure

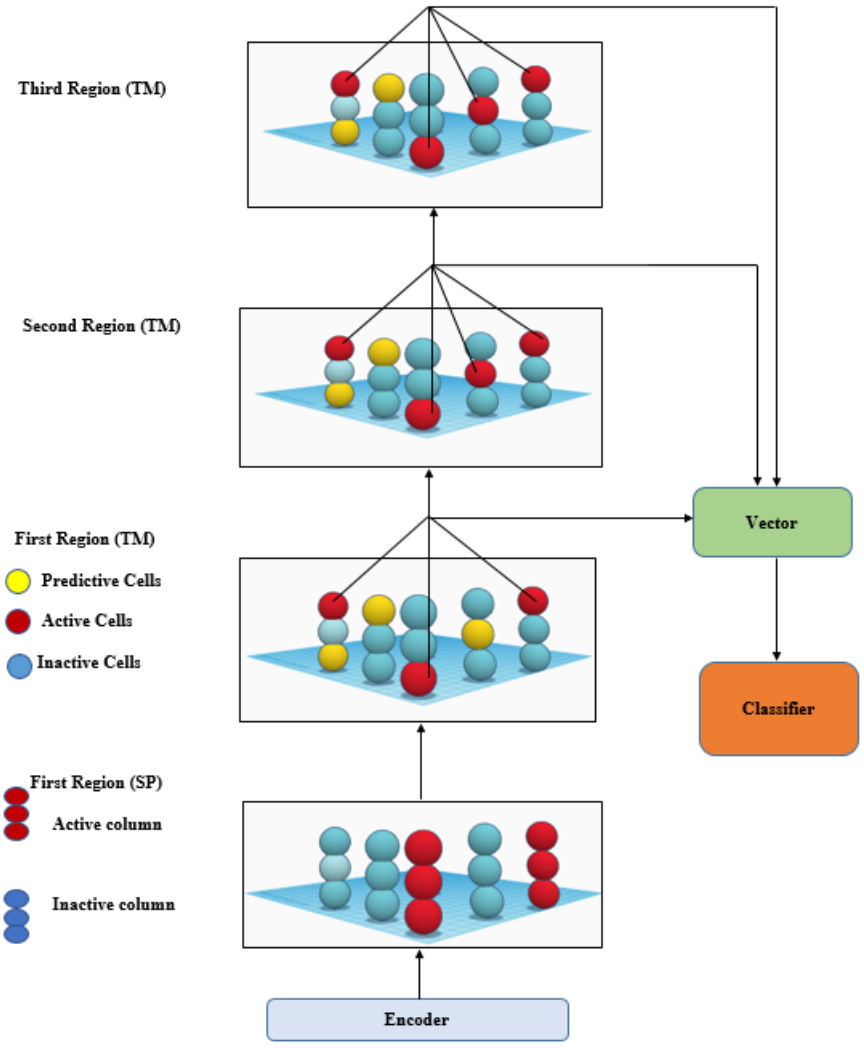


FIGURE 5.7: Cascaded Temporal Memories Stream.

5.4.8 Summary

The proposed technique of using two or more regions that allow learning input patterns from streaming datasets provides a more comprehensive representation of patterns because it used more than the temporal memory in the hierarchy. Every region contains a certain functionality, and the more regions that are increased in the hierarchy, the hierarchy is able to predict patterns more robustly. The author proposed and used the two novel multi-region CLA techniques that include a hash encoder and an MLP classifier to improve the prediction performance in smart home and compared the results with those for one region. For the results of using the proposed technique, see the chapter 6.

Chapter 6

Test and Evaluation

6.1 Introduction

This Chapter introduces descriptions of procedures, evaluation methodology and experiment design. Moreover, the next sections present the preparation of the datasets and the performance metrics used to evaluate the proposed models. This Chapter provides the results of applying the base-line CLAs, the proposed techniques of CLA as well as state-of-the-art supervised learning techniques for activity classification in the smart home.

6.2 Evaluation Methodology

This research is divided into two phases. The first phase aims to apply and evaluate the CLA performance to identify its weaknesses and limitations in the smart home domain. The second phase aims to modify the CLA to overcome the identified weaknesses in the first phase.
The evaluation of machine learning algorithms is limited by the lack of real datasets from smart homes. Due to the high cost of building real smart home datasets, there is a need for powerful simulation tools that can represent the ADLs of the inhabitants. These simulation tools offer flexibility, scalability and accessibility for the researcher (Alshammari *et al.*, 2017; Bouchard *et al.*, 2010).

To evaluate the CLA and other state-of-the-art machine learning algorithms performance, this research will use real and simulated smart home datasets. The following subsections describe the preparation of the datasets and the performance metrics used to assess the quality of the proposed models.

6.2.1 Real Smart Home Dataset:

For the real-world dataset, ARAS dataset was used (Alemdar *et al.*, 2013a). The dataset was generated from real activities engaged by multi-inhabitants in two separate smart homes during a period spanning two months. The dataset contains 20 columns of binary data that represent the sensor values (0 represents OFF state, and 1 represents ON state) sampled each second; whereby, column 21 represents the activity labels for Resident 1, and column 22 represents the activity labels for Resident 2. The residents perform various labelled activities (27 different activities).

This research used this dataset but has combined the activities into five categories, namely: Other, Sleeping, Eating, Personal, and Relaxing. Resulting in a total collection of 25 activities. Another pre-processing step was performed to reduce the size of the dataset from 86400 records per day to 1,440 records. The reduction was done by changing the sample rate from seconds per day to minutes per day. The reduction took care of the sensors' values that changed in-between minutes by retaining these changes. This step reduced the overall dataset size from 5,184,000 records to 86400 records. The motivation behind this step came after the analysis of the dataset which revealed that most of the transitions of activities take longer than one minute.

6.2.2 Synthetic Smart Home SIMADL Dataset

This research used OpenSHS (Alshammari *et al.*, 2017) which is an open source simulation tool that offered the flexibility needed to generate the inhabitants data for classification of ADLs. OpenSHS was used to generate several synthetic datasets that includes 29 columns of binary data representing the sensor values; whereby, column 30 represents the activity labels for each inhabitant, and column 31 represents the timestamp. The sampling was done every second. Seven participants were asked to perform their simulations using OpenSHS. Each participant generated six datasets resulting in forty-two SIMADL datasets in total. The participants self-labelled their activities during the simulation. The labels used by the participants were: Personal, Sleep, Eat, Leisure, Work, Other. The total number of records is 2,674,910 records.

6.2.3 Experiment Design

After the preparation of the forty-two SIMADL synthetic datasets and the two real-world datasets, the records of each dataset are fed to a machine learning model. For each dataset, the data was split into two parts: a training part, where the model is learning

from the data without scoring its performance, and a testing part, where the model's classification accuracy is evaluated. The training part size is 80% of the total size of the dataset and the remaining 20% is used for testing. No shuffling was performed because the activities have natural sequence progression. All the sensor's readings were fed to the tested models with the exception of the timestamp column. The timestamp column was not included as one of the input features because of the inability for some of the models to work with this data type and to ensure a levelled playing ground for all the evaluated models.

6.2.4 Performance Metrics

The performance of the HTM model generated by the standard NuPIC was quantitatively measured to evaluate its effectiveness, through metrics depending on the evaluation criteria. Examples of evaluation metrics are Precision, Recall and F-Measure. The same datasets and evaluation procedure will be used with state-of-the-art machine learning algorithms e.g. AdaBoost, CNN, DTs, HMM, LSTM, MLP, SVM, SGD and SP in order to compare their performance with the proposed technique. This research uses the same two datasets, the real and the simulated smart home datasets, to test and evaluate the performance of the novel approaches, presented in Chapter 5 are using the same evaluation metrics. The novel multi-region CLA techniques that include a hash encoder and an MLP classifier are based on the CLA but with modification for the first region, encoder due to the limitations of the standard NuPIC encoder when dealing with multidimensional datasets. In this research creates a custom encoder, a novel modification for the encoder region so that the data can be fed to the new encoder (a Hash encoder). Instead of using standard NuPIC encoders.
This research has created a predictive model that encompasses massive data, including number of activities (labels). In order for the model to correctly predict each activity, it requires evaluating the performance of the model by metrics, depending on the evaluation criteria. The evaluation metrics used are Precision, Recall and F-Measure.

The inhabitants' ADLs usually vary from one inhabitant to another. Moreover, the ratio of the performed activities for each inhabitant is usually not similar. For example, the "Sleeping" activity may constitute a bigger portion of the whole activity space. Thus, it is good to assume that the labels are imbalanced (Ordóñez & Roggen, 2016). Therefore, simple metrics such as the accuracy metric, shown in equation 6.1, are not suitable for this type of data.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

A naive classifier that predicts the labels with highest frequencies could get high accuracy score. To overcome this issue, F1 score (F-Measure) was used. As shown in equation 6.2, F1 score is defined in terms of Precision and Recall.

$$F1 = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{6.2}$$

The precision is the ratio of the relevant points that have been selected by the model to the total selected points, as shown in equation 6.3, where *TP* is the *True Positives* and *FP* is the *False Positives*.

$$precision = \frac{TP}{TP + FP} \tag{6.3}$$

The recall is the ratio of relevant points that have been selected by the model to the overall total of the relevant points, as show in equation 6.4, where *TP* is the *True Positives* and *FN* is the *False Negatives*.

$$recall = \frac{TP}{TP + FN} \tag{6.4}$$

The percentage of wrong classifications (PWC) is the ratio of the wrongly classified classes to the total testing classes.

$$PWC = \frac{100 * (FN + FP)}{TP + FN + FP + TN} \tag{6.5}$$

6.2.5 Parameter Selection for the CLA Model

The first phase is swarming, the basic method used to find a set of optimized parameters of the model for a certain dataset. The NuPIC framework is unable to determine the nature of the data types and structure; thus, using the swarm algorithm is required to identify and select the best model parameters to be configured before feeding the data into the CLA. Additionally, it is used to identify the encoder type. The second phase is training, in which passing each record from the dataset to the CLA model will enable learning for certain records from the dataset. The third phase is testing, in which passing a portion of the dataset to the CLA model with learning in a disabled state.

6.3 Results

In this section presents the overall comparison results of various models and identifies the overall accuracies associated with each dataset over the supervised machine learning techniques, as well as the CLA. These results and statistics obviously show each algorithm's performance for activity classification with each dataset. This section can be divided into two parts: first part is base-line CLAs versus proposed multi-region CLA techniques results and second part is state-of-the-art machine learning techniques results.

6.3.1 Base-line CLAs Versus Proposed Multi-Region CLA Techniques.

This research used NuPIC version 1.0.3 Algorithms API[1] and NuPIC Bindings version 1.0.0 to implement classification for the inhabitants daily living activities. The models were abbreviated as shown in Table 6.1. As shown in Figure 6.4 the components of each proposed model were presented. Figures in (6.1, 6.2, 6.3) and (Appendix B Tables of Results) show a summary of classification' results this experiment illustrates the performance of novel multi-region CLA techniques that combine a hash encoder and an MLP classifier on the performance of activity classification techniques and the percentage of correct classification. Moreover, these results tested the CLA using the standard NuPIC encoders and SDR classifier on smart home datasets.
This study was compared the CLA with the standard NuPIC encoders and SDR classifier against the CLA with hash encoder and SDR or MLP classifier , as well as two or three regions with a hash encoder and MLP classifier. CLA3 achieved 66.38% F-measure on the real dataset from House A, and the highest F-measure score, 92.63%, was achieved in CLA2 model on the real dataset from House B. In the synthetic SIMADL dataset (Average), a classification F-measure score of 96.21% was obtained with the CLA3 model.

The base-line CLAs with the standard NuPIC encoders did not produce good results. The performance of the CLA could have been improved if a custom and more suitable encoder was used. The high-dimensional and binary nature of the data types in the datasets was a challenge for the existing encoders. The existing encoders are designed to work with simple scalar and categorical datatypes.

This research used a new hash encoder for activity classification in a smart home. Its performance was compared with the CLA with the standard NuPIC encoders; CLA0 achieved a prediction rate in House A of F-measure 61.74%, while the prediction rate in House B was represented by 88.91% F-measure score. In the synthetic SIMADL dataset

[1] http://nupic.docs.numenta.org/1.0.3/index.html

(average), the achieved F-measure by the CLA0 was 89.38%, and it showed better results than the CLA with the standard NuPIC encoders.

The new hash encoder was used with an SDR classifier in one region (CLA0). Its performance was compared with the CLA classifier instead of the SDR classifier using the same model parameters. The CLA with the SDR classifier produced meaningful predictions, providing better results than the CLA with the CLA classifier.

The CLA2, CLA3 were competitors in House A, and the forty-two datasets with the state-of-the-art machine learning techniques. The CLA2 model exceeded the state-of-the-art machine learning techniques in House B with an F-measure of 92.63%.

The CLA1 model has achieved better results than the CLA0, and the CLA1 model used a hash encoder with an MLP classifier instead of an SDR classifier in one region. The MLP classifier could produce meaningful predictions for different ADL patterns.

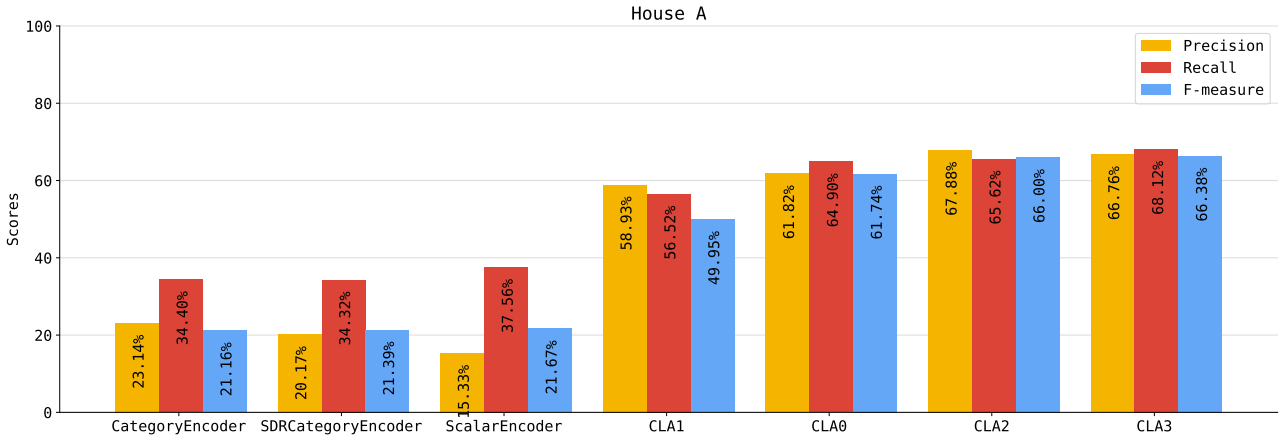


FIGURE 6.1: Comparison between different CLA settings and the proposed techniques-House-A.

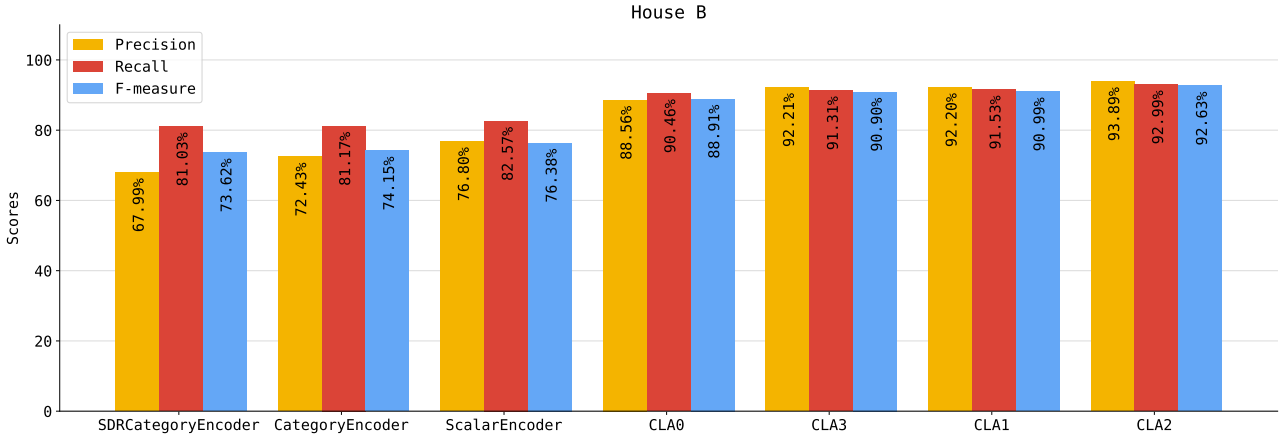


FIGURE 6.2: Comparison between different CLA settings and the proposed techniques-House-B.

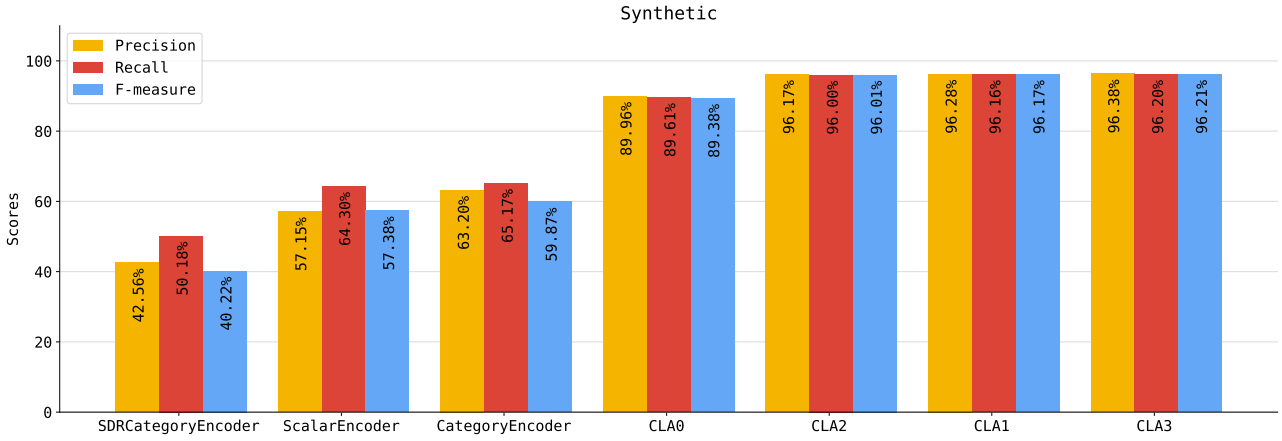


FIGURE 6.3: Comparison between different CLA settings and the proposed techniques-SIMADL dataset.

TABLE 6.1: Abbreviations for the evaluated models

SDRCategoryEncoder	CLA with SDR Category Encoder and SDR Classifier
CategoryEncoder	CLA with Category Encoder and SDR Classifier
ScalarEncoder	CLA with Scala Encoder and SDR Classifier
CLA0	CLA one region with Hash encoder and SDR Classifier.
CLA1	CLA one region with Hash encoder and MLP Classifier.
CLA2	CLA two region (Parallel Stream) with Hash encoder and MLP Classifier.
CLA3	CLA three region (Cascaded TMs) with Hash encoder and MLP Classifier.

Models	**Hash encoder**	**MLP Classifier**	**Two regions CLA**	**Three regions CLA**
CLA0	√			
CLA1	√	√		
CLA2	√	√	√	
CLA3	√	√		√

FIGURE 6.4: The components of each proposed model.

The results shown in Figures (6.1, 6.2 and 6.3) show different findings. It is obvious that the obtained results for the real datasets showed less performance accuracy than the synthetic datasets. Because the inhabitants are asked to assign their activities manually, it was prone to human errors. The real-world datasets have noisy data, such as missing data and data from faulty sensors. Especially, the real-world dataset ARAS House A is inconsistent because one of the inhabitants left the house for long period of time,

however this has not been shown in the dataset, instead House A shows continuous activities, which affects the performance results. The CLA0 used a new hash encoder to deal with the high-dimensional binary data, which explains the increase in performance over the standard NuPIC encoders, as the standard NuPIC encoders are only able to deal with a small number of columns. In addition, CLA1 applied the MLP classifier rather than the SDR classifier. The MLP classifier receives the SDR output from the TM to produce better predictions. Hence, the MLP classifier outperformed the classifiers used in the current implementation of CLA. Furthermore, CLA2 has used the multi-region CLA to remove the limitation of memory management of the original CLA. CLA2 can learn both short-term and long-term patterns in parallel. In the one-region CLA, the model learns either short-term or long-term patterns, not both of them. CLA2 settings have improved the long-term and short-term prediction performance in the context of a smart home. In addition, CLA3 uses a three-region CLA, where the first region learns smaller features, and the second and third regions are more abstract in order to learn and recognise patterns, this allows the CLA3 to produce better performance than the one-region CLA.

6.3.1.1 The selected model parameters by the swarm optimiser.

The swarm optimiser was applied to find the best parameters for these models under consideration for both real and synthetic datasets, you can see the suggested model parameters are shown below

```
SpatialPooler(
inputDimensions=(4517,),
columnDimensions=(2048,),
synPermConnected=0.1,
synPermActiveInc=0.05,
synPermInactiveDec=0.1,
globalInhibition=True,
numActiveColumnsPerInhArea=40,
maxBoost=1.0,
potentialPct=0.8),

TemporalMemory(
columnDimensions = (2048,),
cellsPerColumn=32,
initialPermanence=0.21,
minThreshold=11,
maxNewSynapseCount=20,
permanenceIncrement=0.1,
permanenceDecrement=0.1,
activationThreshold=14,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32)
```

LISTING 6.1: The suggested model parameters by the swarm optimiser.

The swarm optimiser was used to export the CLA models' parameters, for Standard NuPIC encoders that contains many types of encoders such as scalar encoder, category encoder and SDR category encoder. However, the swarm did not generate good models parameter as shown in Listing 6.1, because the dataset has many sensors.

```
SpatialPooler(
inputDimensions=(600,),
columnDimensions=(600,),
synPermInactiveDec= 0.0000081,
synPermActiveInc=0.05,
synPermConnected=0.1,
globalInhibition=True,
numActiveColumnsPerInhArea=38,
boostStrength=1.0,
potentialPct=0.8,

TemporalMemory(
columnDimensions = (600,),
cellsPerColumn= 4,
initialPermanence=0.21,
minThreshold=4,
maxNewSynapseCount=20,
permanenceIncrement=0.1,
permanenceDecrement=0.1,
activationThreshold=9,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32,
```

LISTING 6.2: The best parameters found for Standard NuPIC Encoders models and SDR Classifier models in synthetic datasets.

Using the model parameters as shown in Listing 6.2, the CLA has a classification rate in the forty-two SIMADL datasets of average F-measure with SDR-Category encoder was 40.22% , the Category encoder achieved 59.87% and the Scalar Encoder achieved 57.38%, although the `columnDimensions`, `inputDimensions` were set =1024 in the Scalar Encoder, to increase the model performance.

Using the same model parameters, the CLA with Hash encoder and SDR Classifier was applied to the synthetic dataset, the average F-measure was 76.05%. Additionally, the CLA with Hash encoder and CLA Classifier was used, the average F-measure was 72.61%.

This research attempted to find the best parameters for these models in both real and synthetic datasets, you can see the suggested model parameters in Section 6.4

```
SpatialPooler(
inputDimensions=(1024,),
columnDimensions=(600,),
synPermInactiveDec=0.0007,
synPermActiveInc=0.0003,
synPermConnected=0.1,
globalInhibition=True,
numActiveColumnsPerInhArea=20,
boostStrength=1.0,
potentialPct=0.95,

TemporalMemory(
columnDimensions = (600,),
cellsPerColumn= 10,
initialPermanence=0.21,
minThreshold=9,
maxNewSynapseCount=20,
permanenceIncrement=0.1,
permanenceDecrement=0.1,
activationThreshold=9,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=22,
```

LISTING 6.3: The best parameters found for Standard NuPIC Encoders models and SDR classifier models in real datasets.

Using the model parameters as shown in Listing 6.3, the CLA with SDR-Category encoder in ARAS, the real dataset has obtained a classification rate in House A of F-measure was 21.39 %, while that for House B was F-measure 73.62%, the Category encoder in House A achieved 21.16 %, while that for House B was 74.15%, the Scalar Encoder in House A achieved 21.67 %, while that for House B was 76.38%. The `columnDimensions` were set to = 700 in the SDR-Category encoder and in the Scalar Encoder the `columnDimensions` were set to =500.

Using the same model parameters, the CLA with the Hash encoder and SDR Classifier was applied to the ARAS, the real dataset, the prediction achieved rate of F-measure in House A was 53.94 %, in House B was 86.71%. Additionally, the CLA with the Hash encoder and CLA Classifier was applied to House A and the achieved, F-measure was 34.50% in House A, and 84.28% in House B.

```
# First Region

SpatialPooler(
inputDimensions=(3400,),
columnDimensions=(600,),
synPermInactiveDec= 0.0000081,
synPermActiveInc=0.05,
synPermConnected=0.1,
globalInhibition=True,
```

```
numActiveColumnsPerInhArea=43,
boostStrength=1,
potentialPct=0.8),

TemporalMemory(
columnDimensions = (600,),
cellsPerColumn= 4,
initialPermanence=0.21,
minThreshold=11,
maxNewSynapseCount=20,
permanenceIncrement=0.001,
permanenceDecrement=0.001,
activationThreshold=12,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32,

# Second Region

SpatialPooler(
inputDimensions=(3400,),
columnDimensions=(600,),
synPermInactiveDec= 0.0000081,
synPermActiveInc= 0.05,
synPermConnected=0.1,
globalInhibition=True,
numActiveColumnsPerInhArea=43,
boostStrength=1.0,
potentialPct=0.8),

TemporalMemory(
columnDimensions = (600,),
cellsPerColumn= 7,
initialPermanence=0.21,
minThreshold=7,
maxNewSynapseCount=20,
permanenceIncrement=0.000000585,
permanenceDecrement=0.000000585,
activationThreshold=9,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32,

# Third Region

TemporalMemory(
columnDimensions = (600,),
cellsPerColumn= 15,
initialPermanence=0.21,
minThreshold=7,
maxNewSynapseCount=20,
permanenceIncrement=0.000000585,
permanenceDecrement=0.000000585,
activationThreshold=9,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32,
```

LISTING 6.4: The best parameters found for novel multi-region CLA techniques that include Hash Encoder models and SDR/MLP classifier models.

After, empirical testing to find the best model parameters for the proposed model, the parameters shown in Listing 6.4 have produced better scores. One of parameters in models' parameters it called w, it represents the number of active bits to the total number of bits, w was selected to be = 12 in all models but in the Parallel Spatio-Temporal Memory Stream model (CLA2) w was = 21, and also the `columnDimensions` were set to = 1200 in the first region and the `columnDimensions` = 1500 in the second region, to produce the best results for house A.

6.3.2 State-of-the-Art Techniques Versus Proposed Multi-Region CLA Techniques.

The results of the selected state-of-the-art machine learning models are obtained using several software packages. Primarily the scikit-learn (Pedregosa *et al.*, 2011) and keras (Chollet *et al.*, 2015) were used. Due to the various configurations used for certain models, the models abbreviated as shown in Table 6.2. Using Precision, Recall, and F-measure as evaluation metrics, the results are shown in Figure 6.5 for House A from ARAS dataset, Figure 6.6 for House B from the same dataset, and Figure 6.7 for the synthetic SIMADL dataset. Tables of Results in Appendix B show a summary of the results classifications.

The overall performance of all the evaluated machine learning algorithms across all datasets using F-measure score is summarised in Figure 6.8. The results show competitive performance of the evaluated algorithms. However, three of the top five algorithms are based on neural networks.

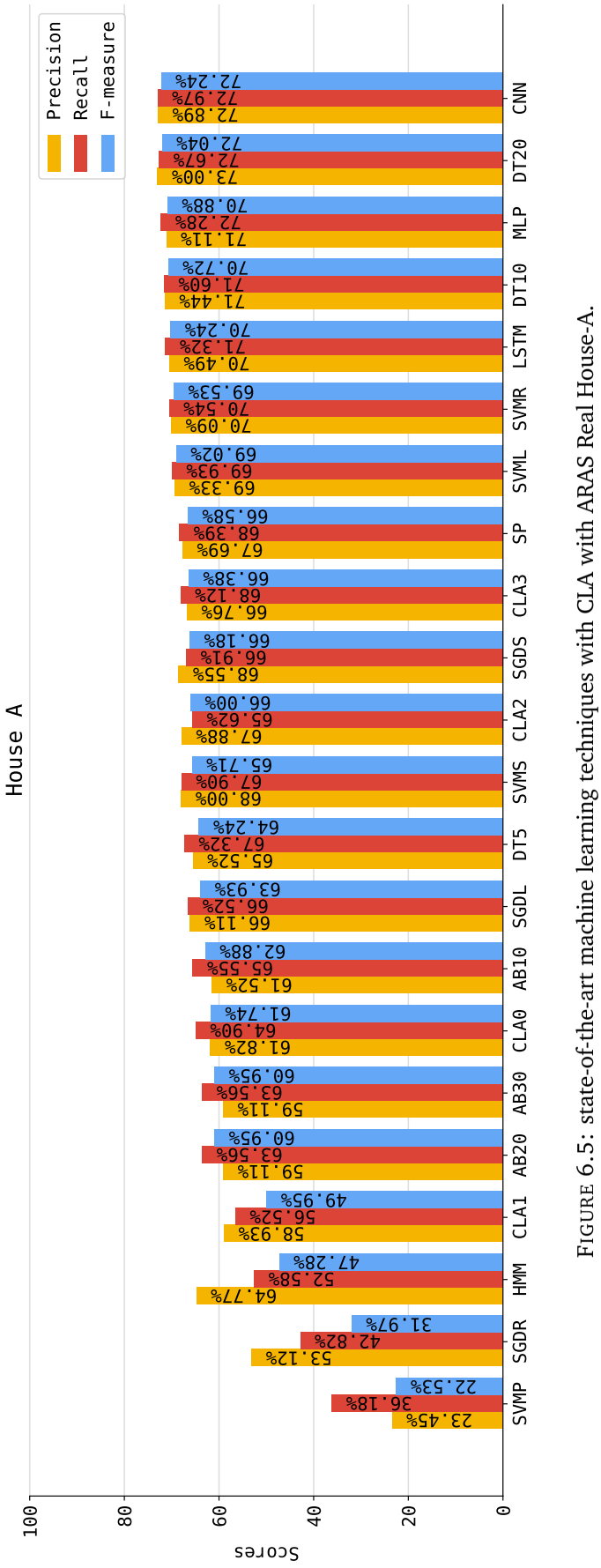


FIGURE 6.5: state-of-the-art machine learning techniques with CLA with ARAS Real House-A.

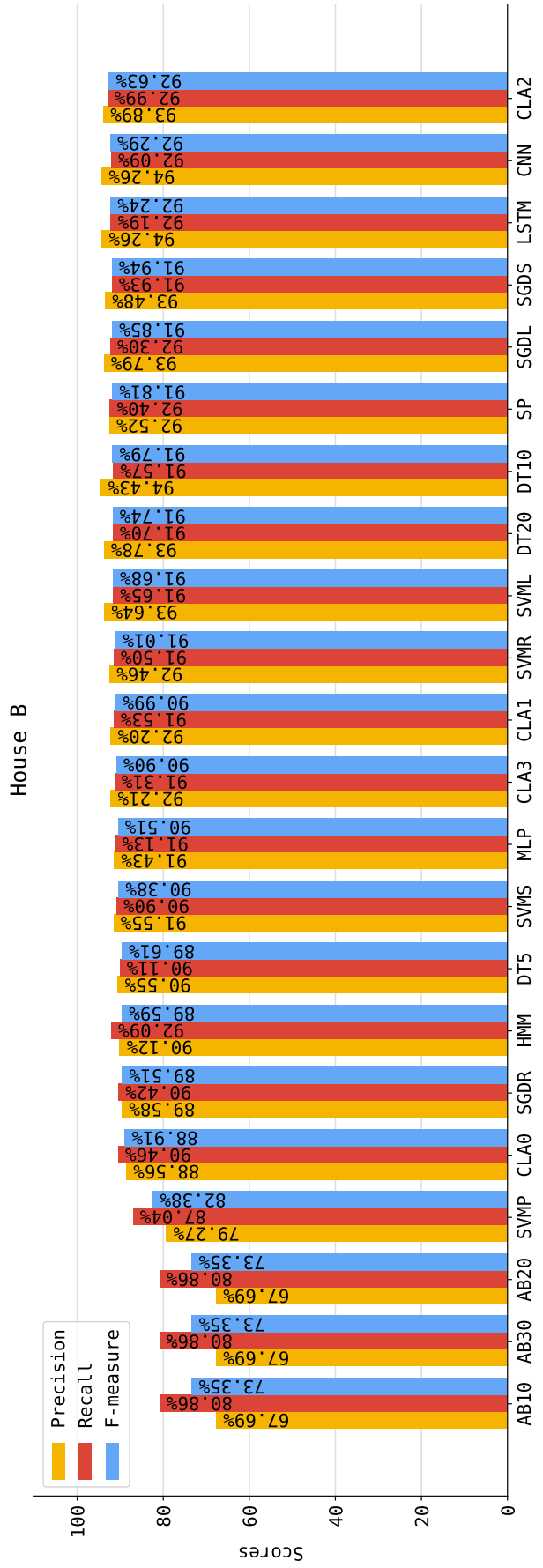


FIGURE 6.6: state-of-the-art machine learning techniques with CLA with ARAS Real House-B.

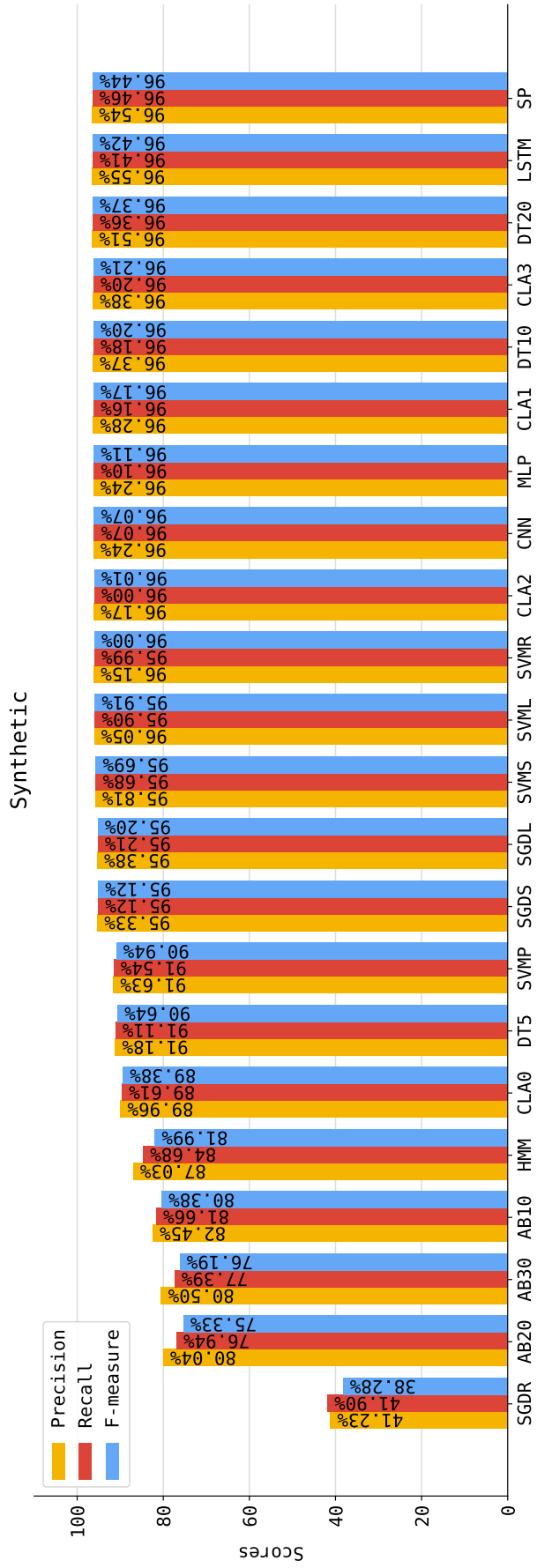


FIGURE 6.7: state-of-the-art machine learning techniques with CLA with Synthetic SIMADL Dataset (Average).

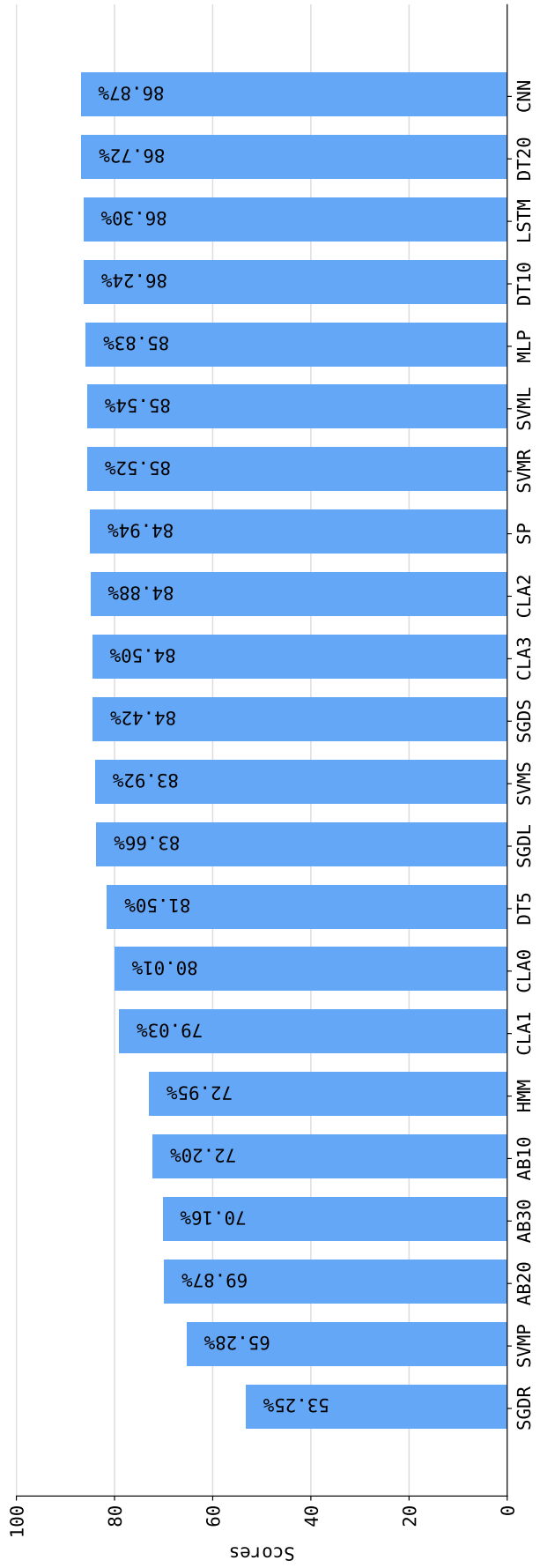


FIGURE 6.8: The average F-measure score across all datasets.

TABLE 6.2: Abbreviations for the evaluated models

AB10	AdaBoost with 10 estimators
AB20	AdaBoost with 20 estimators
AB30	AdaBoost with 30 estimators
CLA0	Cortical Learning Algorithm one region with Hash encoder and SDR Classifier.
CLA1	CLA one region with Hash encoder and MLP Classifier.
CLA2	CLA two region (Parallel Stream) with Hash encoder and MLP Classifier.
CLA3	CLA three region (Cascaded TMs) with Hash encoder and MLP Classifier.
CNN	Convolutional Neural Network
DT5	Decision Tree with max depth of 5
DT10	Decision Tree with max depth of 10
DT20	Decision Tree with max depth of 20
HMM	Hidden Markov Model
LSTM	Long Short Term Memory
MLP	Multi-layer Perceptron
SVMR	Support Vector Machine with RBF kernel
SVML	Support Vector Machine with linear kernel
SVMP	Support Vector Machine with polynomial kernel
SVMS	Support Vector Machine with sigmoid kernel
SGDS	Stochastic Gradient Descent with linear SVM function
SGDR	Stochastic Gradient Descent with regression function
SGDL	Stochastic Gradient Descent with logistic regression function
SP	Structured Perceptron

6.4 Discussion

After evaluating the performance of each algorithm for activity classification with each dataset, the results for different models are shown in the figure 6.5, 6.6 and 6.7. The results of applying the proposed algorithm and using the hash encoder (CLA0 model) instead of the standard HTM encoders seemed promising, and it improved the performance of the model as well as the CLA1, which model used an MLP classifier in one region. The CLA1 achieved better results than the CLA0 that used a SDR classifier. However, recently, state-of-the-art algorithms have produced better results than both the CLA0 and CLA1 models. This research used two novel multi-region CLA techniques that include a hash encoder and an MLP classifier (CLA2 algorithm and CLA3 algorithm) to improve the long- and short-term prediction performance in the context of a smart

home. The results illustrate the influence of using two novel multi-region CLA techniques on performance for activity classification. Both novel techniques (CLA2 model and CLA3 model) are able to compete with and surpass the performance of the state-of-the-art algorithms on some datasets.

The choice of the kernel type used with SVM proved to be crucial. Using a polynomial kernel produced the worst results especially in House A. While using a linear kernel and an RBF kernel produced the best results for the SVM algorithm. Increasing the number of estimators for AdaBoost did not improve the results and the best results obtained with ten estimators. The algorithms based on neural networks showed superior performance over most other algorithms. However, the CLA2 achieved the highest F-measure (92.63%) of all prediction models for the second house (B) of the real-world, ARAS dataset. DT, LSTM, SVM and SGD are good candidates for the task at hand. While HMM, AdaBoost and SGD with regression function showed much less performance accuracy. HMM as an unsupervised algorithm does not compete well with other supervised algorithms in this context.

In another research effort, the HMM was evaluated on the ARAS dataset and the reported accuracy results for House A is 61.5% and for House B is 76.2% (Alemdar *et al.*, 2013a). The proposed techniques obtained accuracy results of 53.9% for House A and 92.3% for House B. It is worth noting that the differences in the results could be attributed to the preformed pre-processing steps. The previously mentioned work used leave-one-out cross validation on the dataset and the reported results were the average accuracy. For House A, the minimum accuracy was 46.3% and the maximum accuracy was 88.4% . For House B, the minimum accuracy was 31.1% and the maximum accuracy was 96.7% (Emi & Stankovic, 2015). In this work no cross-validation technique was used because the assumption was that the activities have natural sequence progression and any cross-validation step will break this assumption.

As expected, the obtained results for the real datasets were more challenging than those for the synthetic datasets.

The CLA results appeared promising. Both real and synthetic datasets used in this study were derived from a smart home environment whether real or simulated. These datasets included multiple binary sensory inputs represented in a multi-column array. The swarm process with the standard NuPIC encoders was not suitable for a large number of features; therefore, it did not provide good model parameters.

This research followed all four rules to apply a new encoder in order to deal with multi-dimensional datasets and create a new encoder. Hence, the CLA performance

has achieved consistently good results and is competitive with other machine-learning algorithms to classify and predict daily activities in a smart home.

(Alemdar *et al.*, 2013a) proposed the application of a HMM to classify the activities of daily living in multiple house inhabitants. They developed the ARAS dataset to assess their proposed algorithm. They achieved a classification rate of 61.5% in House A, while that for House B was 76.2% (Alemdar *et al.*, 2013a). (Prossegger & Bouchachia, 2014) used decision trees to classify activities of daily living in multiple house inhabitants. The application was called E-ID5R. They used the same ARAS dataset to assess their proposed algorithm. The evaluation of the classication of the activities in House A has a classication rate of 40%, while that for House B is around 82%. The CLA2 and CLA3 have classication rates in House A and House B that were better than their proposed algorithms in (Alemdar *et al.*, 2013a), (Prossegger & Bouchachia, 2014). However, decision tree algorithms have achieved good results after a pre-processing step was performed on the ARAS dataset.

Other studies (Lavesson & Davidsson, 2006), (Domingos, 2012) indicated that the model parameters play an important role in the performance of the algorithm. Sometimes, the algorithm is suitable for this task, but if the selection of the model parameters is inappropriate, the performance will be low. This research used particle swarm optimisation to generate the best model parameters, as shown in Listing 6.1, but the swarming step did not produce the best model parameters, and there is a need to make some changes and update the model parameters to improve the prediction accuracy, as shown in Listing 6.4.

6.5 Summary

In this chapter, the evaluation methodology and experiment design were illustrated. Moreover, it described the preparation of the datasets and the performance metrics used to evaluate the machine-learning technique. Forty-two ADL datasets were generated from the OpenSHS tool, and ARAS real-world datasets were used to test and evaluate the quality of the models. It provided more information concerning the results and findings of the supervised learning technique for classification of ADLs including tables and figures of results. Every experiment started reading each dataset separately one by one and was fed to the classification model. The dataset was split into 80% for training the model to learn from the data without scoring, and 20% was used for testing data in the model with performance scoring. This research found the parameters for these models in both real and synthetic datasets and suggested the best parameters for CLA

models. The results revealed that the CLA performance with the standard NuPIC encoders was not good compared with the state-of-the-art supervised learning techniques for classification of ADLs. Both novel multi-region CLA techniques demonstrated good performance compared with the state-of-the-art machine-learning techniques and they achieved the best performance than base-line CLAs. The Parallel Spatio-Temporal Memory Stream model (CLA2) model achieved F-measure for House (A) from the ARAS dataset was 66.00%, the highest scores F-measure 72.24% was achieved by the CNN. The CLA2 model achieved F-measure of 92.63% for House (B), which was the best of all prediction models. The average F-measure for the synthetic SIMADL datasets 96.01% was achieved the CLA2, the highest F-measure scores 96.44% was achieved by the SP. performance of both novel multi-region CLA techniques is promising.

Chapter 7

Conclusions and Future Work

7.1 Summary

The Internet of things (IoT) paradigm includes many applications. The smart home is one of the most prominent parts of the IoT paradigm. Smart homes contain several sensors, devices, home appliances, electronics, etc. These sensors can generate a huge volume of data. Machine-learning techniques can make use of such data to enable an occupant to automate the control and management of these sensors and devices for healthcare, elderly care, home energy management, security and safety applications, etc. Effective machine-learning techniques are required to perform data integration and learning of ADLs from sensor readings. Such techniques are used to predict and classify the inhabitant‘s activities. There are many types of machine-learning techniques: supervised, unsupervised, and reinforcement learning. Some techniques use probabilistic and statistical methods, and others use sequence-learning algorithms. In a smart home environment, machine-learning techniques need to adapt to the inhabitant‘s changing habits. Hence, this research has adapted the use of supervised learning techniques.

Various machine-learning techniques have been implemented in smart home environments. However, the literature review of the classification of ADLs showed a lack of a comprehensive evaluation of such machine-learning algorithms in this domain. This work first conducted a review of the current research in predicting and classifying ADLs and then performed an evaluation of the current state-of-the-art supervised learning techniques for activity classification in the smart home domain. This study has shown competitive performance of evaluated techniques. Moreover, the best results of the state-of-the-art techniques in this domain are achieved by techniques based on neural networks such as CNN, SP and LSTM.

To propose machine-learning techniques for the classification of ADLs, there is a need to capture representative datasets that can be used to train and test the machine-learning models and to evaluate their performance. The literature review of prediction and classification of ADLs in smart homes indicated that there is a lack of real-world datasets due to the excessive cost of building real smart homes. This research used the OpenSHS an open-source, cross-platform 3D smart home simulator for dataset generation. Additionally, forty-two datasets, SIMADL, were generated using the simulation tool OpenSHS for the simulation of ADLs in a smart home, where seven participants simulated their ADLs for different contexts.

This work conducted an assessment of the CLA with the standard NuPIC encoders and an SDR classifier for activity classification in a smart home. The performance of the CLA revealed the weakness of using the standard NuPIC encoders with multi-columnar datasets. Because the dataset in a smart home includes multiple binary sensors, a new encoder is required to deal with high-dimensional sensors.

The results of using the hash encoder instead of the standard NuPIC encoders appear promising. However, state-of-the-art algorithms produced better results than the CLA with the hash encoder. This research proposed two novel multi-region CLA techniques that comprise a hash encoder and an MLP classifier to improve the long- and short-term prediction performance in the context of a smart home. The MLP classifier was used to produce a better prediction of the inhabitant‘s activities in a smart home environment.

The proposed two novel multi-region CLA techniques have been compared against base-line CLAs, and state-of-the-art algorithms. The Parallel Spatio-Temporal Memory Stream model (CLA2) achieved average F-measure of 84.88% across all datasets, and the Cascaded Temporal Memories Stream (CLA3) scored F-measure of 84.50% across all datasets. While the best performance of base-line CLAs achieved an average F-measure of 51.80% over all datasets. Thus, the proposed techniques showed a 33.08% increase in performance over the base-line CLA.

The results show that the base-line CLA performance was not good compared with the state-of-the-art supervised learning techniques for classification of ADLs. the proposed technique (CLA2) showed 1.99% decrease from the best results of the state-of-the-art techniques that was achieved by the CNN. Although, in House B of the real-world ARAS dataset, the CLA2 model has achieved the highest F-measure score of all state-of-the-art supervised learning techniques.

7.2 Original Contributions

The original contributions to knowledge of this research project, ordered by importance, include the following:

- Two novel techniques that learns user's patterns and allow the home user to obtain predictions based on the HTM theory and CLA, this has been achieved by proposing two novel multi-region CLA techniques that include a hash encoder and an MLP classifier. A new hash encoder is applied as a modification of the CLA to cope with multiple sensor inputs from a smart home. Because of the existing encoders of the standard NuPIC encoders, such scalar and categorical datatypes are prepared to deal with a single column or a small number of columns. The MLP classifier is implemented rather than the classifiers used in the current implementation of CLA to produce better predictions. Moreover, the development of two novel multi-region CLA techniques aims to solve the issue of the learning input patterns from streaming datasets in smart homes with the one-region CLA, which is restricted. the original CLA learns using one memory level, such model learns either short-term or long-term patterns, not both of them. To remove the limitation of memory management of the CLA, it uses more regions in the CLA with a decrease in the number of cells per column to learn short patterns in the first region. In the second region, the number of cells per column was increased to learn long patterns to solve the issue. This research proposes multi-region CLA techniques to learn short and long-term patterns.

- Forty-two SIMADL datasets are generated using the simulation tool (OpenSHS) for ADL classification problems that simulates the activities taken place in a smart home environment. The participants can interact and classify many different patterns of activities in the virtual environment. OpenSHS captures and records the state of all smart devices and sensors every second. The datasets can be used to evaluate and test machine-learning models, which carry out activity classification tasks in the smart home domain. The datasets are publicly available online for the research community to test and validate their models and any new emerging techniques.

- This research conducts a review of existing research in prediction and classification of ADLs and carries out a comprehensive evaluation of state-of-the-art machine-learning technique applications in the context of smart homes.

7.3 Conclusion

This research reviewed the IoT literature, smart-home applications, and machine-intelligence algorithms typically used in the smart-home context. The literature review was analysed to evaluate the requirements for machine-intelligence techniques to classify events within the context of smart homes.

This research proposed two techniques to predict and handle the classification of activities undertaken by inhabitants living in a smart home and presented the validation and accuracy of the performance of these techniques on smart home data. This research introduced the concept of machine learning with a focus on the area of smart homes, moving through the concept and ideology of HTM as a means to predict a temporal behaviour.

The real-world datasets targeted at classification in the context of smart homes scarcely exist, and few real datasets are available in the public domain. A simulation tool should be used rather than building real smart homes because of the time constraints and effort for the researcher and participants to generate datasets. This work completed a review of existing simulation tools and approaches and contributed to the usability study for the OpenSHS tool. To validate and evaluate the accuracy of machine-learning techniques for classification of ADLs in a smart home there is a need for a good representative dataset, there is a lack of real-world datasets due to the excessive cost of building real smart homes, forty-two SIMADL datasets for classification of ADLs were generated using a simulation tool (OpenSHS). Seven participants simulated their ADLs for different contexts. The participants' opinions about the OpenSHS usability were captured using the SUS. The limitation is that the average age of the participants was 30 and to reuse the same for elderly people scenario, there is a need to repeat the experiments using different participants.

Smart homes have various layouts, and the inhabitants perform their activities in various ways. The same activity might use a completely different sensor activation, depending on the inhabitants changing habits. This variability represents a challenge for this research. To deal with this variability, this research uses supervised learning techniques that can cope with each individual smart-home environment.

Simulation techniques were researched, analysed, and synthesised to produce a best fit for this research. This research used the ARAS dataset, which is captured from the real-world activities of residents in two houses and a synthetic dataset that is generated by the OpenSHS simulation tool. In the real-world dataset, the proposed techniques results show that the predictive effectiveness appears good, although there are many

inhabitant activities in the dataset. In addition, the CLA results appear promising. The output from the dataset needed to be encoded, the memory management of the CLA model is required to learn short and long-term patterns from streaming datasets for classification of ADLs. Various methods were considered, and suitable solutions were derived. From these investigations, this research was conducted using the methods and tools discussed, generating a set of results that were analysed. These, in turn, supported the hypothesis.

The literature review revealed that such machine-learning algorithms lack a comprehensive evaluation in this domain. This work conducted an evaluation of state-of-the-art machine learning techniques and their application in the context of smart homes. The best results of the state-of-the-art techniques was achieved by the CNN.

HTM theory and its implementation, the CLA, have been studied. This work applied the HTM as its CLA has the capability to integrate and learn patterns from streaming datasets and to find meaningful spatio-temporal relations in high-dimensional data. The HTM and its CLA fulfil the requirement for the machine-learning algorithm to classify and predict events in smart homes. The application conducted an assessment for activity classification in a smart home using the HTM theory and its CLA implementation. The evaluation assessed the performance of the CLA that identified weaknesses and limitations, mainly due to the fact that the data generated from the smart home have high-dimensional binary data. The researcher applied a new encoder that is more appropriate for these data. The researcher used an MLP classifier instead of the classifiers used in the NuPIC frame-work to improve the prediction accuracy using a multilayer neural network. Two novel multi-region CLA techniques were developed to learn short-term and long-term patterns from streaming datasets.

The results have shown that the base-line CLA performance with standard NuPIC encoders was not competing with the state-of-the-art machine learning techniques across all datasets. However, the proposed techniques have improved the CLA performance by using two novel multi-region CLA techniques. Two novel multi-region CLA techniques were developed to cope with the problems discovered when using the original CLA to classify ADLs in the smart home, the proposed techniques (CLA2 and CLA3) can compete and outperform the performance of the state-of-the-art algorithms on some datasets.

7.4 Future Work

- Due to the nature of the data in a smart home that includes several types of data, some sensors and devices produce decimal/numeric values, such as temperature

sensors and power consumption sensors, and some others can be used as binary sensors for two states, on and off. For future work, a new encoder could be created for the CLA to learn input of different types of data generated from multiple sensors in a smart home, converting it to one unique SDR, and to deal with high-dimensional binary and non-binary data .Additionally, a new classifier could be developed that is biologically inspired, which can learn a sequence of different types of data generated from multiple sensors in a smart home to infer and predict various ADL patterns.

- To evaluate the performance of machine-learning techniques for activity classification in the context of smart homes, an adaptation to the changing habits of the inhabitant is required, e.g. sometimes the same activity label is selected with various sensor activations based on the changing habits of the inhabitants. To test the machine-learning techniques, datasets must be generated, which include complex resident patterns. More datasets that explore new complex inhabitant patterns are needed, these can be simulated using OpenSHS. For future work, the effects of using different hyperparameters and the resulting accuracy of the model could be investigated.

- OpenSHS can also be extended to incorporate simultaneous activities, this would allow the creation of datasets generated from multiple occupants for classification of ADLs. Instead of binary sensors deployed in the current OpenSHS implementation, researchers can add different kinds of sensors and devices, such as temperature sensors, pressure sensors, etc. OpenSHS can be also used to visualise interactivly the performance of the machine learning algorithms as well as smart home designs. This visualisation would allow researchers to identify drawbacks in a smart home environment. This will help accelerate the development and proposition of new effective designs. Moreover, within the IoT paradigm, the contributed datasets will be used to test and validate different IoT frameworks.

Appendix A

Datasets

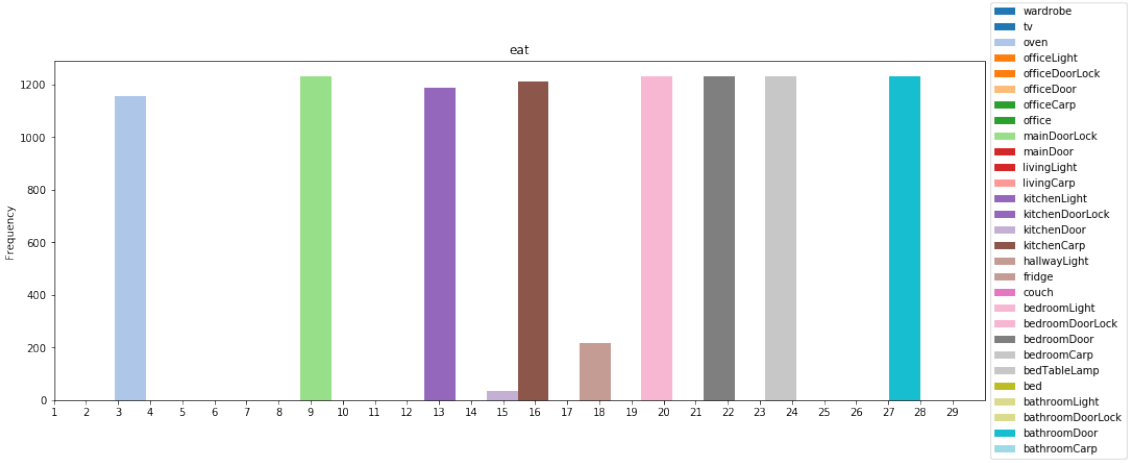


FIGURE A.1: The sensor readings for the Eat activity in the training sample d1-1m-0tm
.

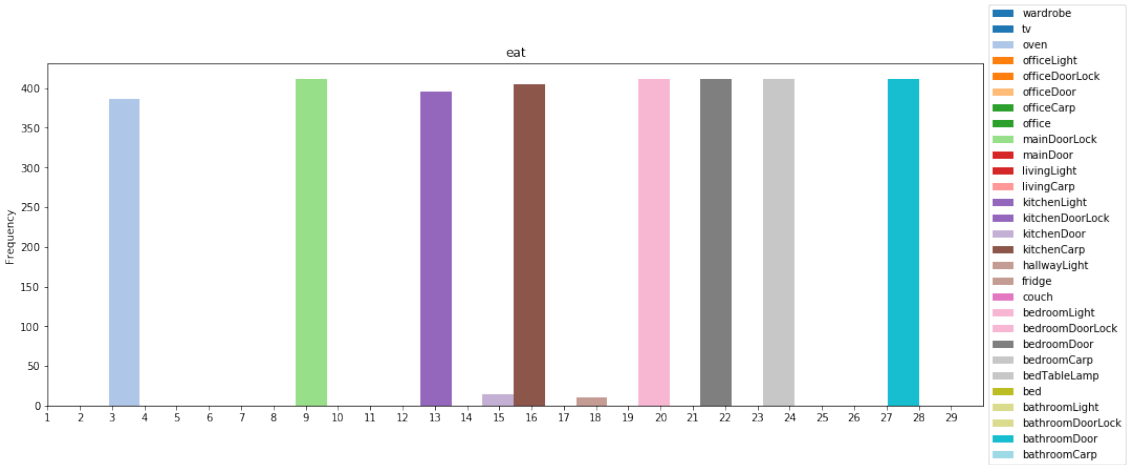


FIGURE A.2: The sensor readings for the Eat activity in the testing sample d1-1m-0tm.

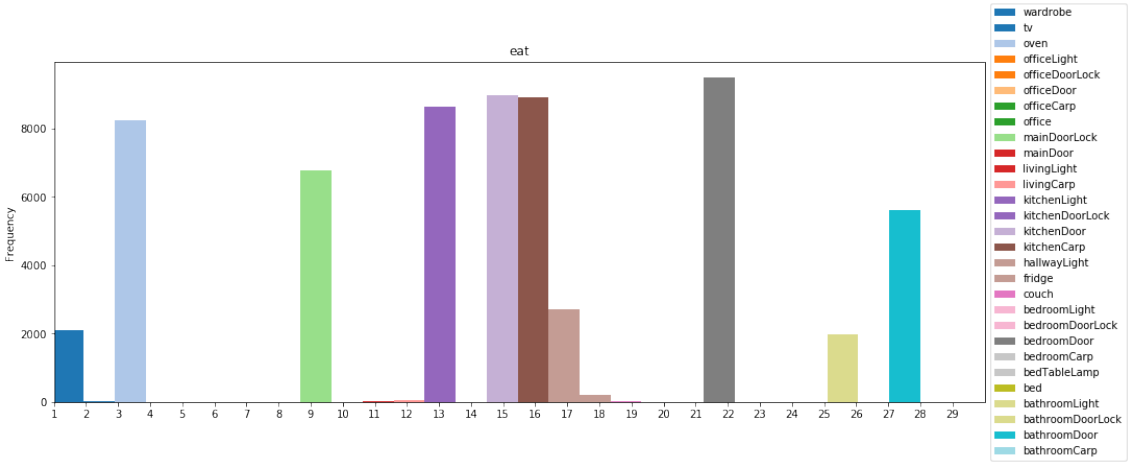


FIGURE A.3: The sensor readings for the Eat activity in the training sample d2-1m-0tm
.

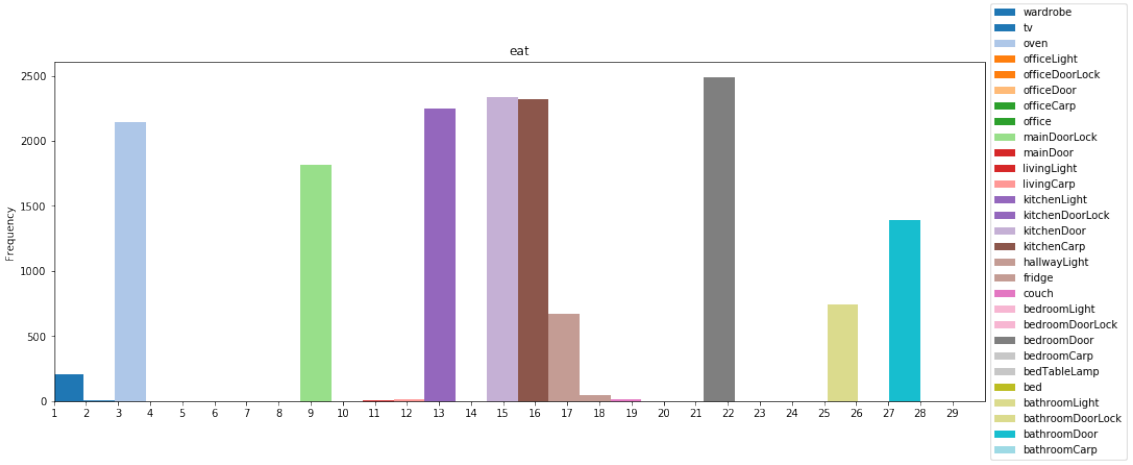


FIGURE A.4: The sensor readings for the Eat activity in the testing sample d2-1m-0tm.

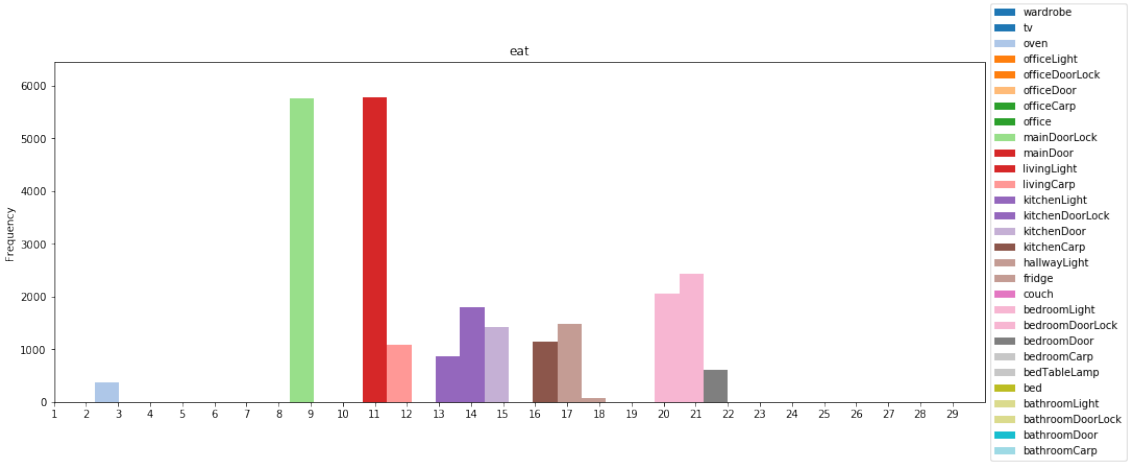


FIGURE A.5: The sensor readings for the Eat activity in the training sample d3-1m-0tm
.

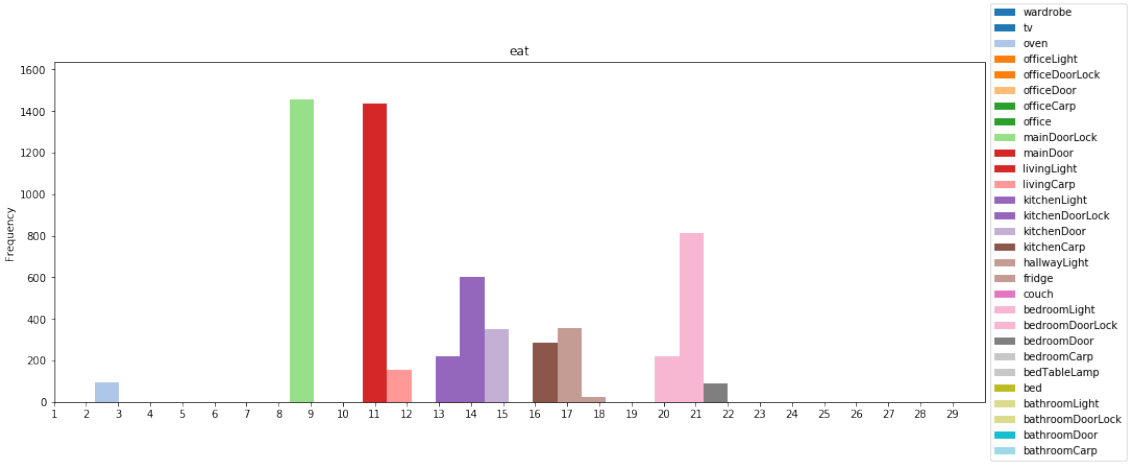


FIGURE A.6: The sensor readings for the Eat activity in the testing sample d3-1m-0tm.

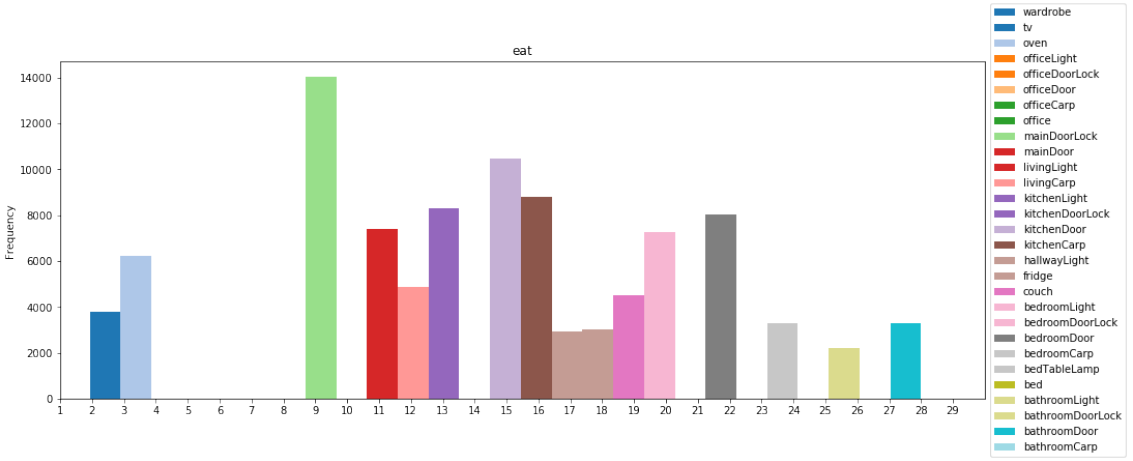


FIGURE A.7: The sensor readings for the Eat activity in the training sample d4-1m-0tm

.

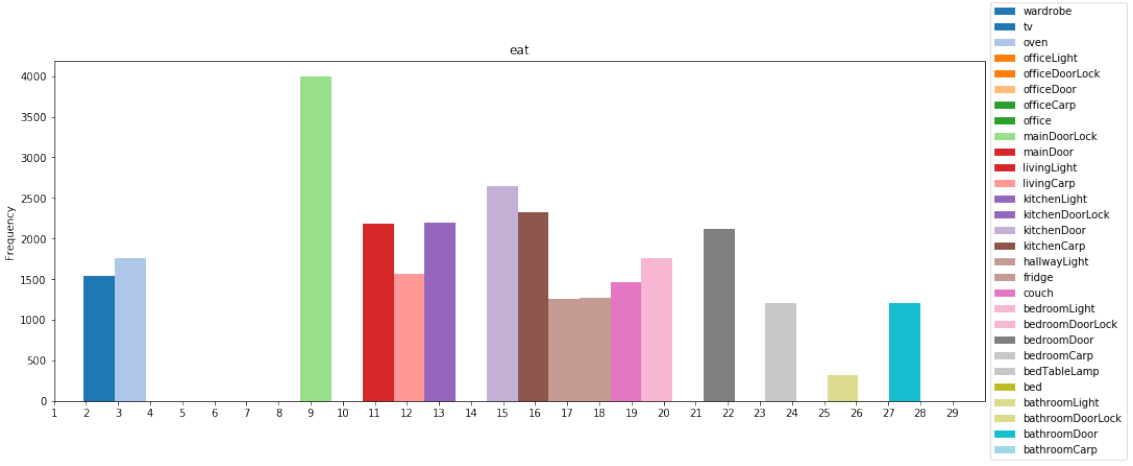


FIGURE A.8: The sensor readings for the Eat activity in the testing sample d4-1m-0tm.

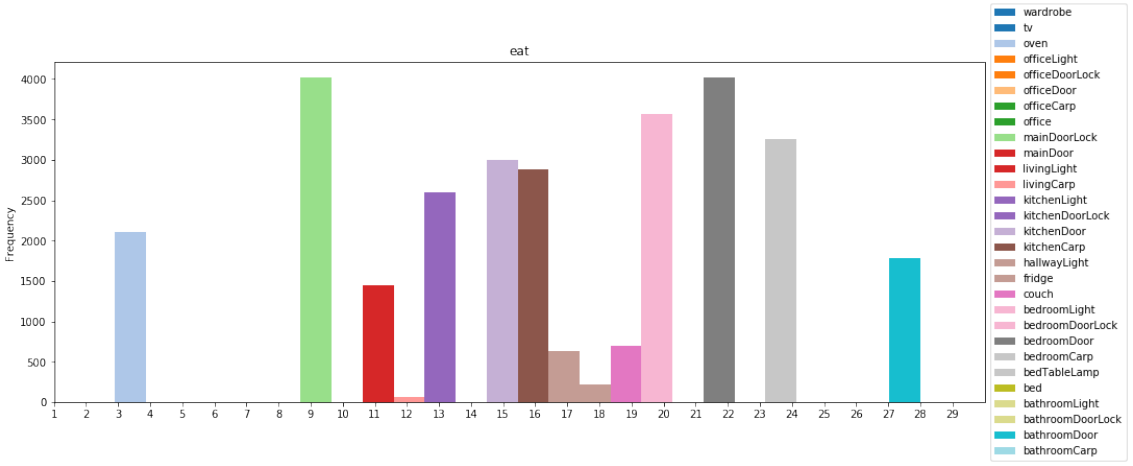


FIGURE A.9: The sensor readings for the Eat activity in the training sample d5-1m-0tm

.

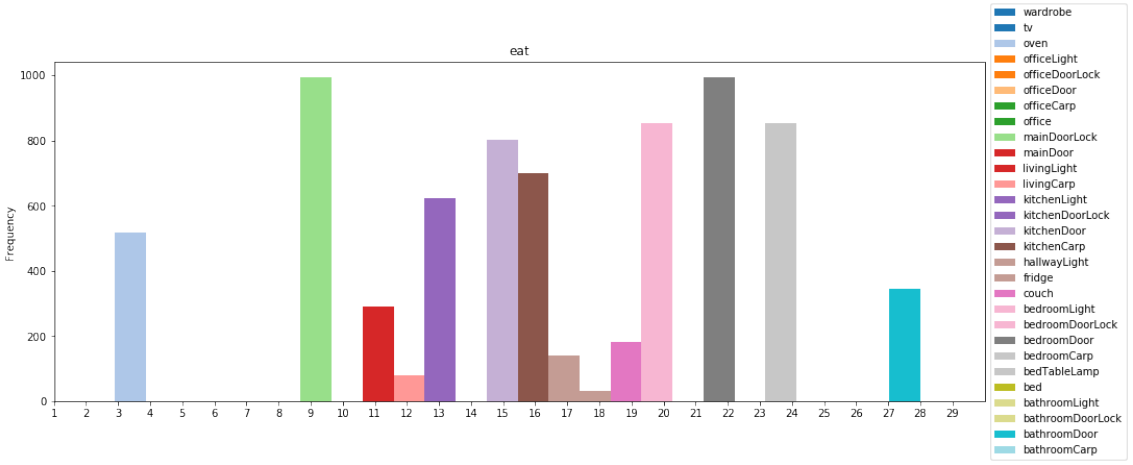


FIGURE A.10: The sensor readings for the Eat activity in the testing sample d5-1m-0tm.

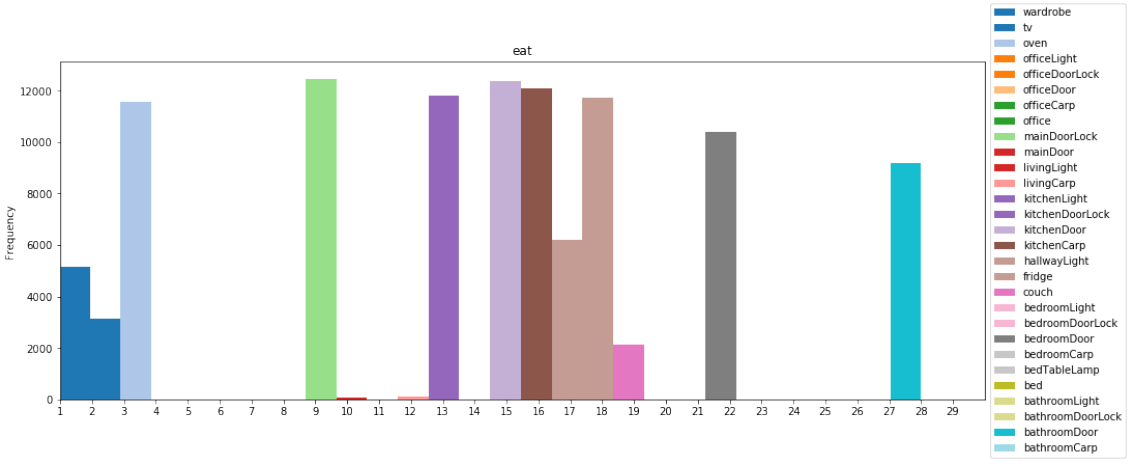


FIGURE A.11: The sensor readings for the Eat activity in the training sample d6-1m-0tm

.

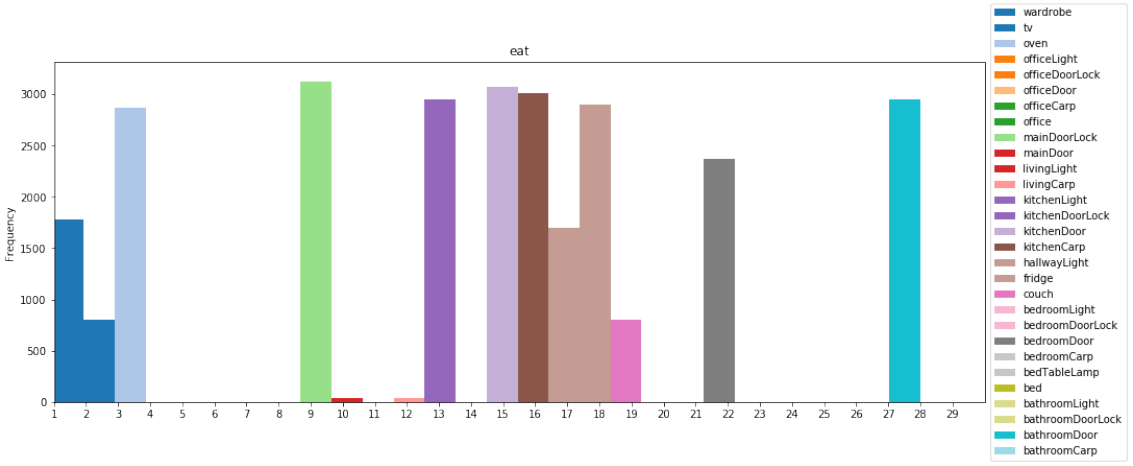


FIGURE A.12: The sensor readings for the Eat activity in the testing sample d6-1m-0tm.

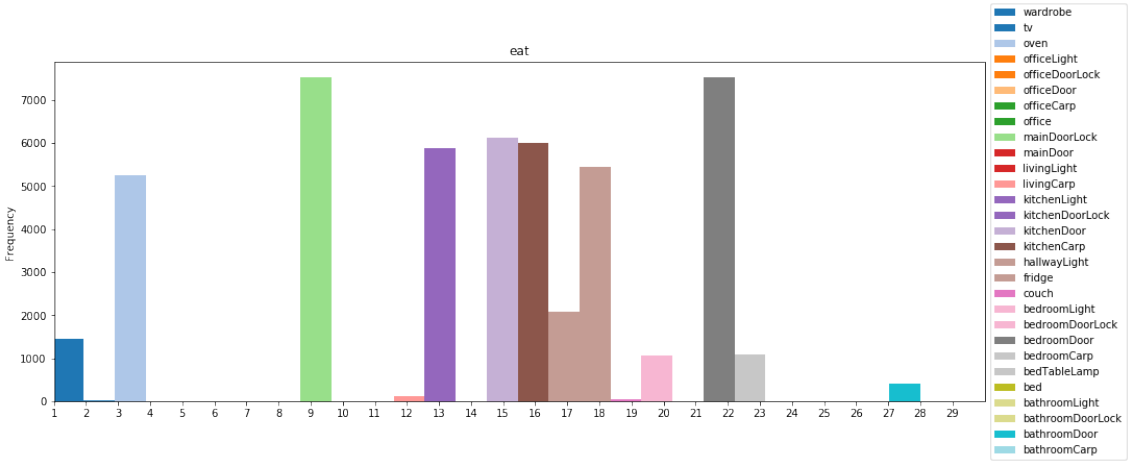


FIGURE A.13: The sensor readings for the Eat activity in the training sample d7-1m-0tm

.

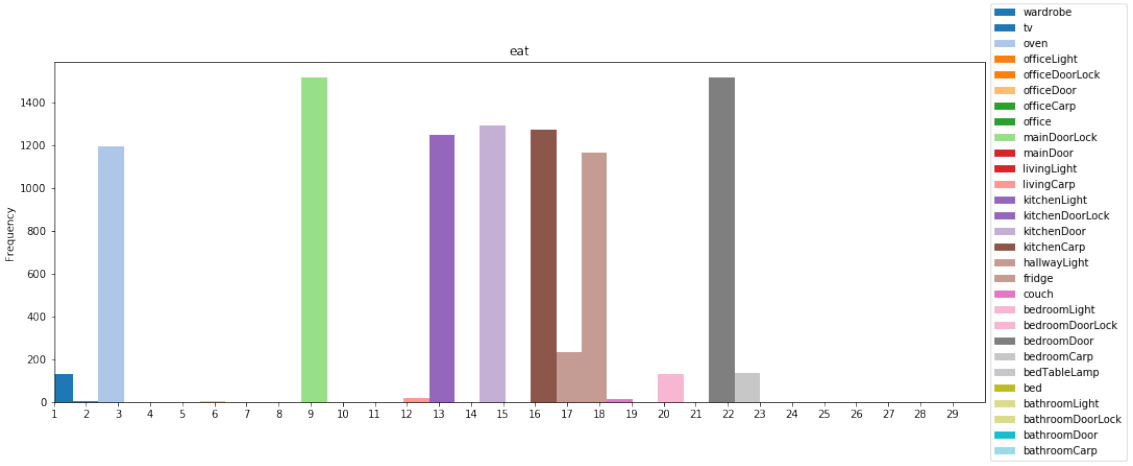


FIGURE A.14: The sensor readings for the Eat activity in the testing sample d7-1m-0tm.

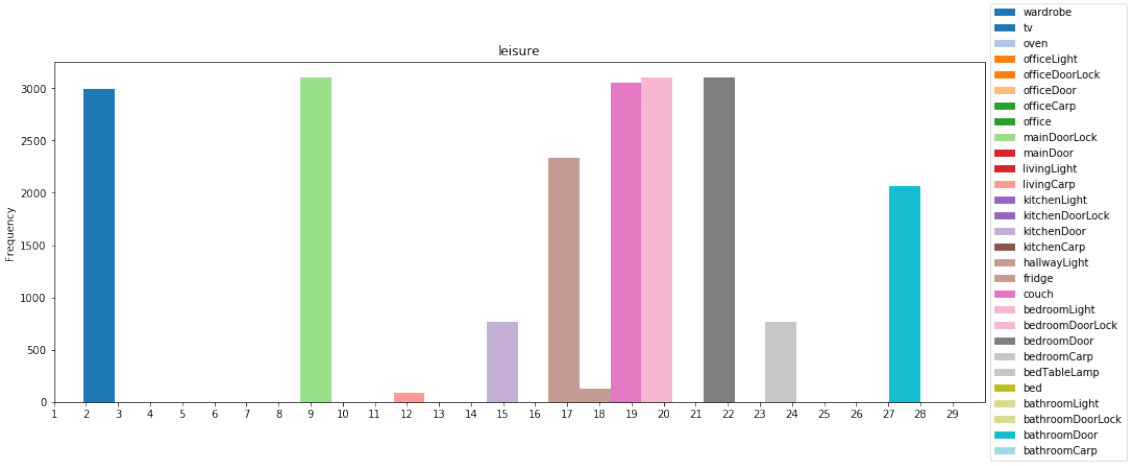


FIGURE A.15: The sensor readings for the Leisure activity in the training sample d1-1m-0tm .

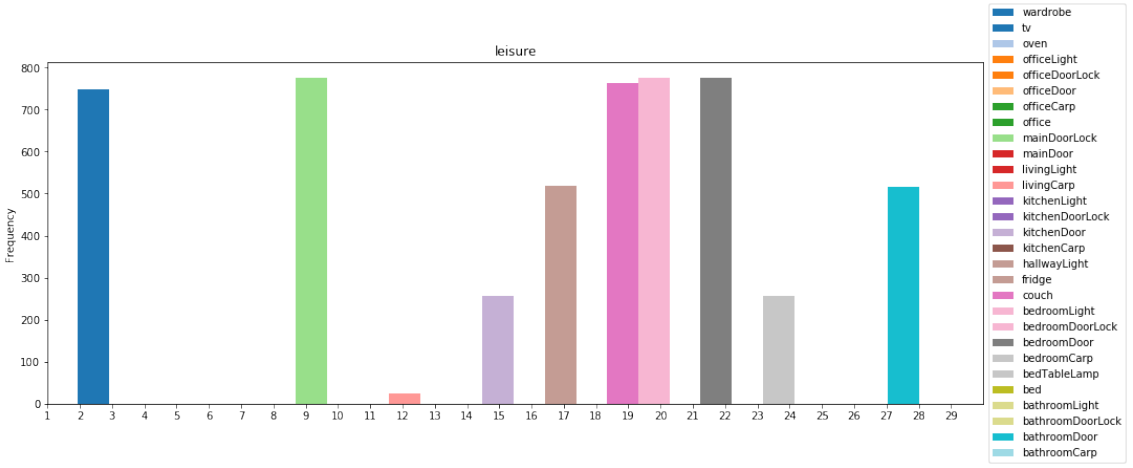


FIGURE A.16: The sensor readings for the Leisure activity in the testing sample d1-1m-0tm.

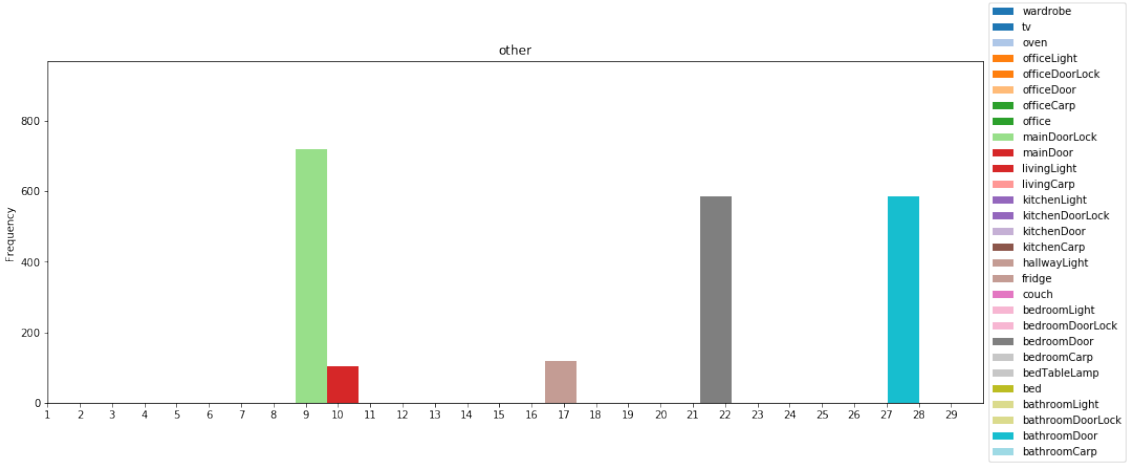


FIGURE A.17: The sensor readings for the Other activity in the training sample d1-1m-0tm .

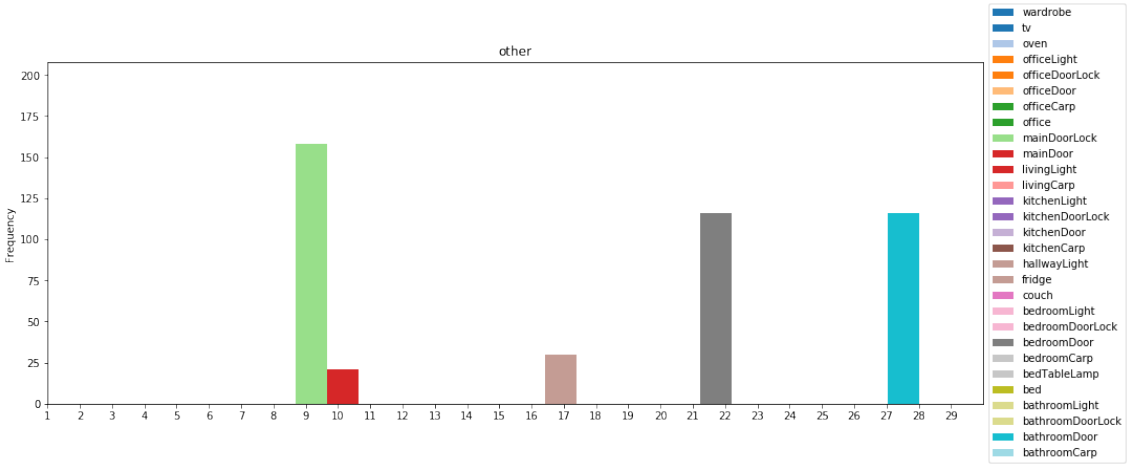


FIGURE A.18: The sensor readings for the Other activity in the testing sample d1-1m-0tm.

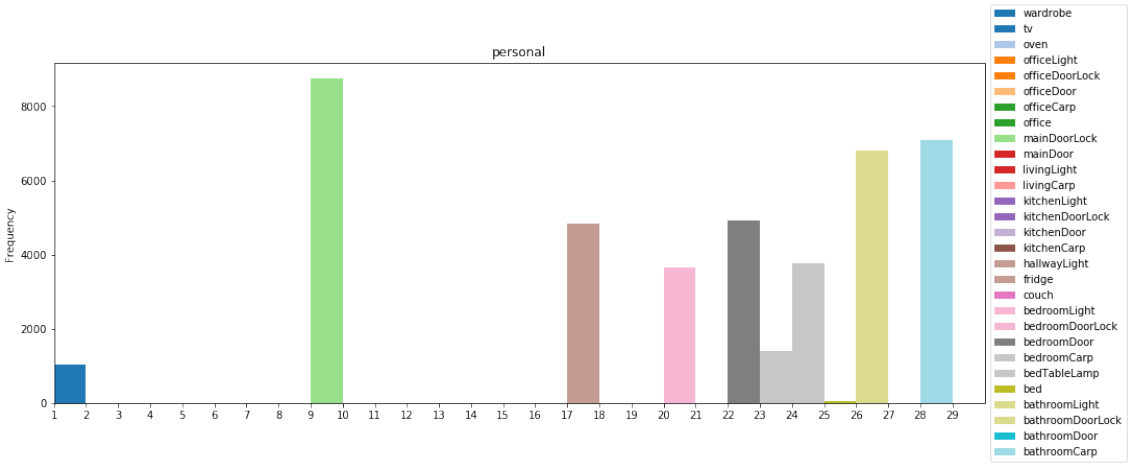


FIGURE A.19: The sensor readings for the Personal activity in the training sample d1-1m-0tm .

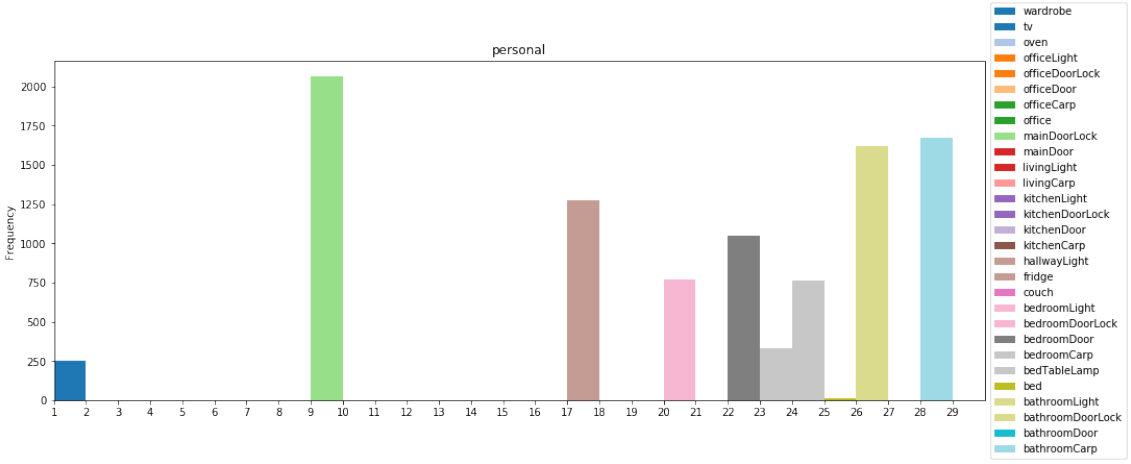


FIGURE A.20: The sensor readings for the Personal activity in the testing sample d1-1m-0tm.

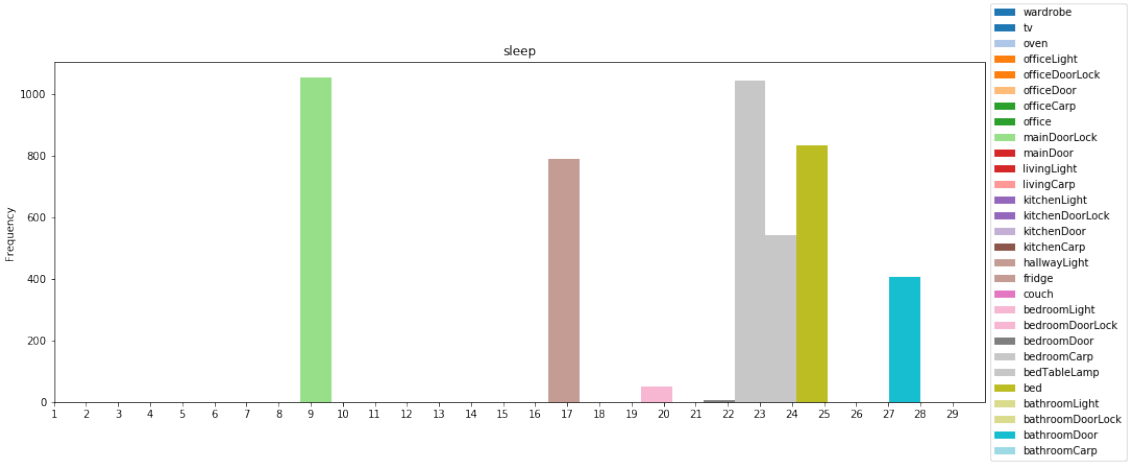


FIGURE A.21: The sensor readings for the Sleep activity in the training sample d1-1m-0tm .

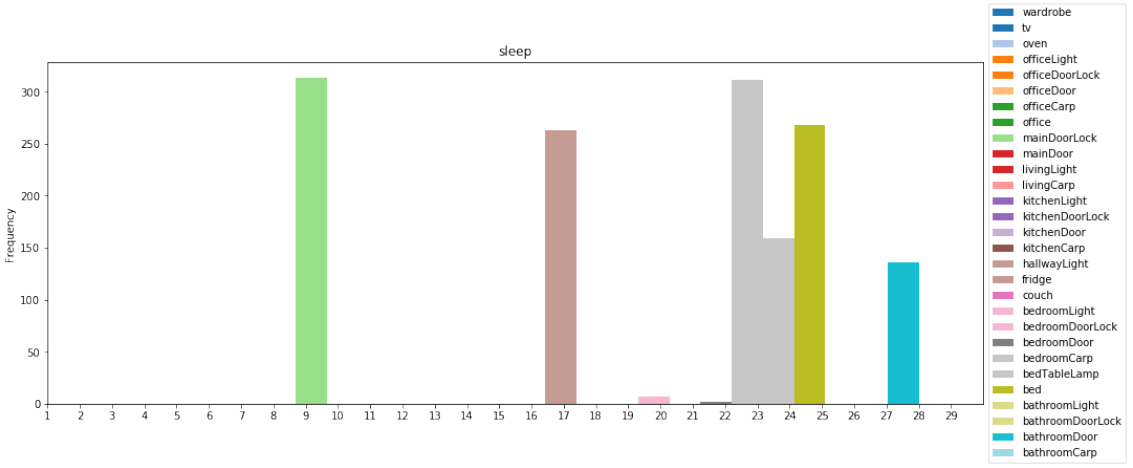


FIGURE A.22: The sensor readings for the Sleep activity in the testing sample d1-1m-0tm.

Appendix B

Tables and Figures of Results

B.1 Tables of Results

TABLE B.1: Results base-line CLAs and proposed techniques ARAS Dataset Real House (A).

Algorithm	Precision	Recall	F-Measure
ScalarEncoder	15.32%	37.56%	21.66%
CategoryEncoder	23.13%	34.40%	21.16%
SDRCategoryEncoder	20.17%	34.32%	21.38%
CLA0	61.82%	64.90%	61.73%
CLA1	58.92%	56.52%	49.94%
CLA2	67.88%	65.62%	66.00%
CLA3	66.75%	68.12%	66.38%

TABLE B.2: Results base-line CLAs and proposed techniques ARAS Dataset Real House (B).

Algorithm	**Precision**	**Recall**	**F-Measure**
ScalarEncoder	76.80%	82.56%	76.37%
CategoryEncoder	72.43%	81.16%	74.14%
SDRCategoryEncoder	67.98%	81.02%	73.61%
CLA0	88.56%	90.46%	88.90%
CLA1	92.20%	91.52%	90.98%
CLA2	93.89%	92.98%	92.63%
CLA3	92.20%	91.30%	90.89%

TABLE B.3: Results base-line CLAs and proposed techniques Synthetic Dataset (Average).

Algorithm	**Precision**	**Recall**	**F-Measure**
ScalarEncoder	57.14%	64.30%	57.38%
CategoryEncoder	63.20%	65.16%	59.87%
SDRCategoryEncoder	42.55%	50.18%	40.21%
CLA0	89.95%	89.61%	89.37%
CLA1	96.27%	96.16%	96.16%
CLA2	96.16%	96.00%	96.01%
CLA3	96.37%	96.20%	96.21%

TABLE B.4: Result state-of-the-art machine learning techniques with CLA - ARAS Dataset Real House (A).

Algorithm	**Precision**	**Recall**	**F-Measure**
SVMR	70.09%	70.54%	69.53%
SVML	69.33%	69.93%	69.02%
SVMP	23.44%	36.18%	22.53%
SVMS	67.99%	67.89%	65.70%
SGDS	68.54%	66.91%	66.17%
SGDR	53.12%	42.82%	31.96%
SGDL	66.11%	66.52%	63.93%
DT5	65.52%	67.32%	64.24%
DT10	71.43%	71.60%	70.71%
DT20	73.00%	72.67%	72.03%
AB10	61.51%	65.54%	62.87%
AB20	59.10%	63.55%	60.94%
AB30	59.11%	63.55%	60.95%
HMM	64.77%	52.58%	47.28%
SP	67.68%	68.39%	66.57%
MLP	71.11%	72.28%	70.88%
CNN	72.89%	72.97%	72.24%
LSTM	70.49%	71.32%	70.24%
CLA0	61.82%	64.90%	61.73%
CLA1	58.92%	56.52%	49.94%
CLA2	67.88%	65.62%	66.00%
CLA3	66.75%	68.12%	66.38%

TABLE B.5: Result state-of-the-art machine learning techniques with CLA - ARAS Dataset Real House (B).

Algorithm	**Precision**	**Recall**	**F-Measure**
SVMR	92.45%	92.45%	91.00%
SVML	93.63%	91.65%	91.67%
SVMP	79.27%	87.03%	82.38%
SVMS	91.54%	90.89%	90.37%
SGDS	93.48%	91.93%	91.94%
SGDR	89.57%	90.42%	89.50%
SGDL	93.78%	92.30%	91.85%
DT5	90.55%	90.10%	89.61%
DT10	94.42%	91.56%	91.79%
DT20	93.77%	91.70%	91.74%
AB10	67.69%	80.86%	73.34%
AB20	67.69%	80.86%	73.34%
AB30	67.69%	80.86%	73.34%
HMM	90.12%	92.09%	89.59%
SP	92.51%	92.40%	91.80%
MLP	91.43%	91.13%	90.5%
CNN	94.26%	92.09%	92.29%
LSTM	94.26%	92.19%	92.24%
CLA0	88.56%	90.46%	88.90%
CLA1	92.20%	91.52%	90.98%
CLA2	93.89%	92.98%	92.63%
CLA3	92.20%	91.30%	90.89%

TABLE B.6: Result state-of-the-art machine learning techniques with CLA Synthetic Dataset (Average).

Algorithm	**Precision**	**Recall**	**F-Measure**
SVMR	96.15%	95.99%	96.00%
SVML	96.05%	95.90%	95.91%
SVMP	91.62%	91.54%	90.93%
SVMS	95.81%	95.68%	95.68%
SGDS	95.33%	95.12%	95.12%
SGDR	41.22%	41.89%	38.28%
SGDL	95.37%	95.21%	95.19%
DT5	91.17%	91.11%	90.64%
DT10	96.37%	96.18%	96.19%
DT20	96.51%	96.36%	96.37%
AB10	82.45%	81.66%	80.38%
AB20	80.04%	76.94%	75.32%
AB30	80.49%	77.38%	76.18%
HMM	87.03%	84.68%	81.99%
SP	96.54%	96.45%	96.44%
MLP	96.24%	96.10%	96.11%
CNN	96.24%	96.07%	96.07%
LSTM	96.55%	96.41%	96.42%
CLA0	89.95%	89.61%	89.37%
CLA1	96.27%	96.16%	96.16%
CLA2	96.16%	96.00%	96.01%
CLA3	96.37%	96.20%	96.21%

B.2 Prediction accuracy with Ground Truth

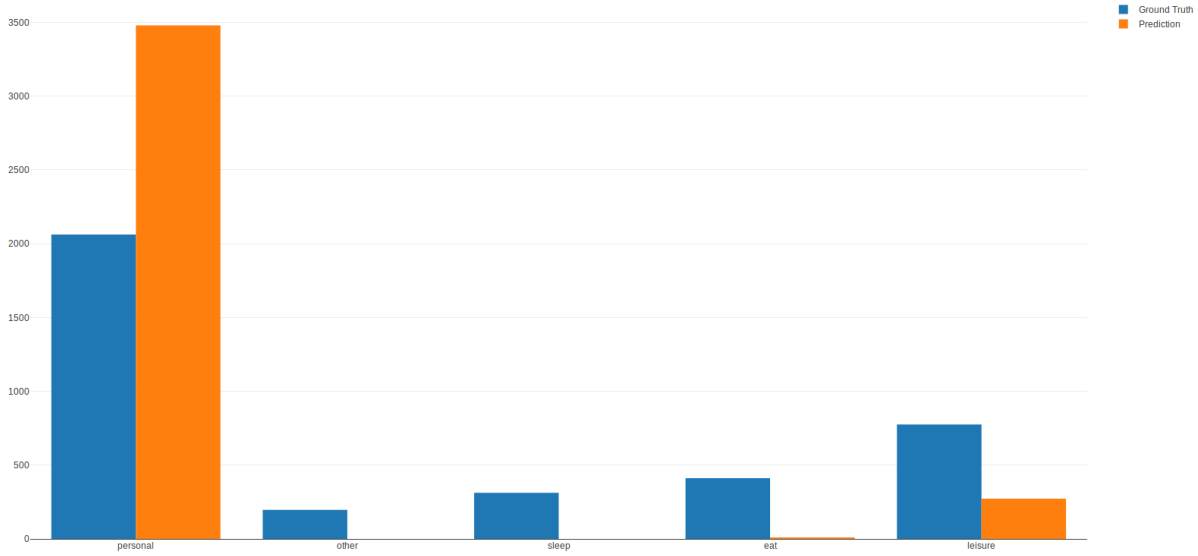


FIGURE B.1: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using ScalarEncoder .

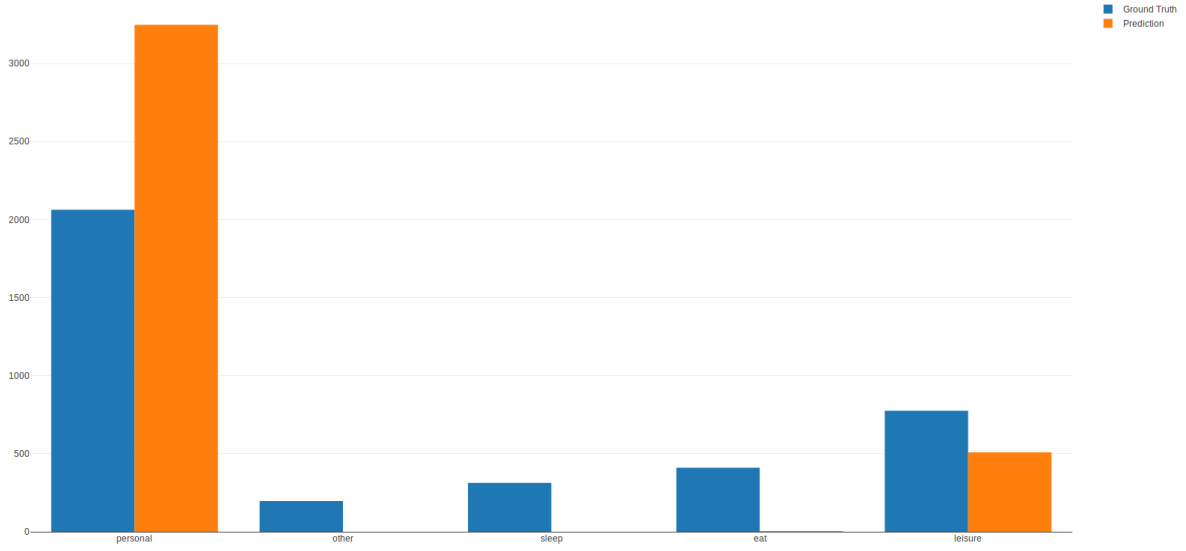


FIGURE B.2: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using CategoryEncoder.

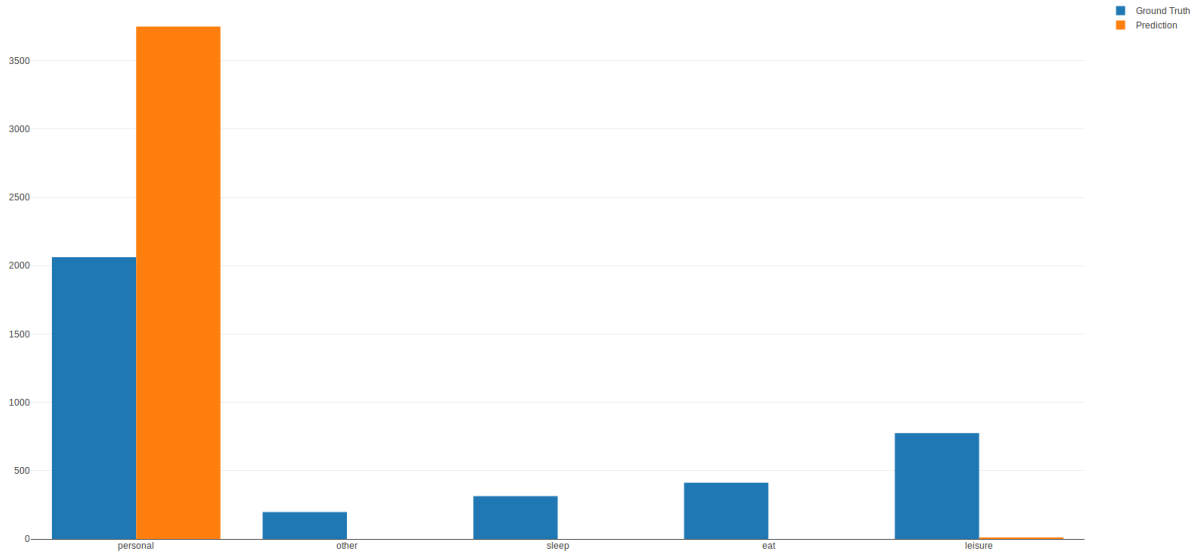


FIGURE B.3: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using SDRCategoryEncoder.

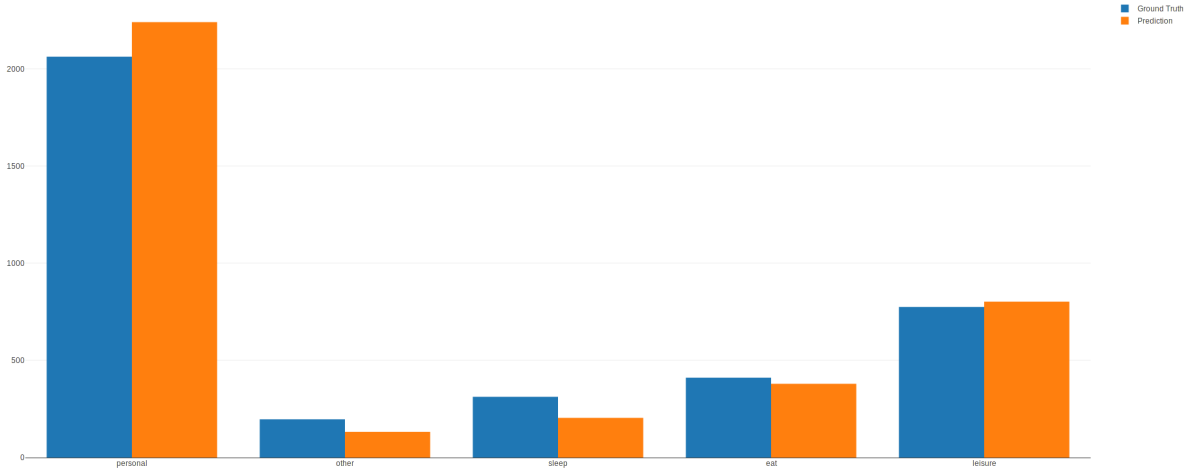


FIGURE B.4: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using CLA0.

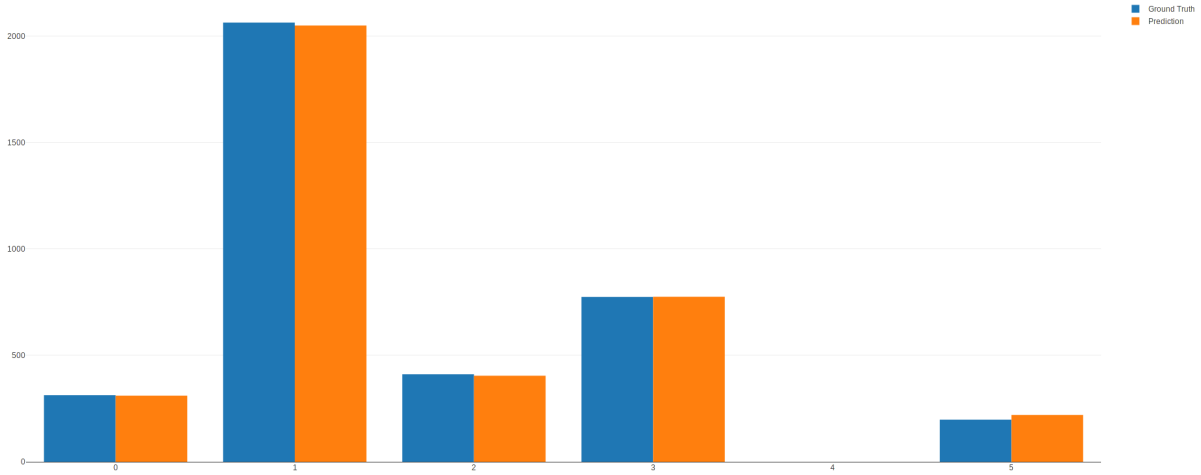


FIGURE B.5: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using CLA1. 'sleep': 0 , 'personal': 1, 'eat': 2, 'leisure': 3, 'work':4, 'other': 5

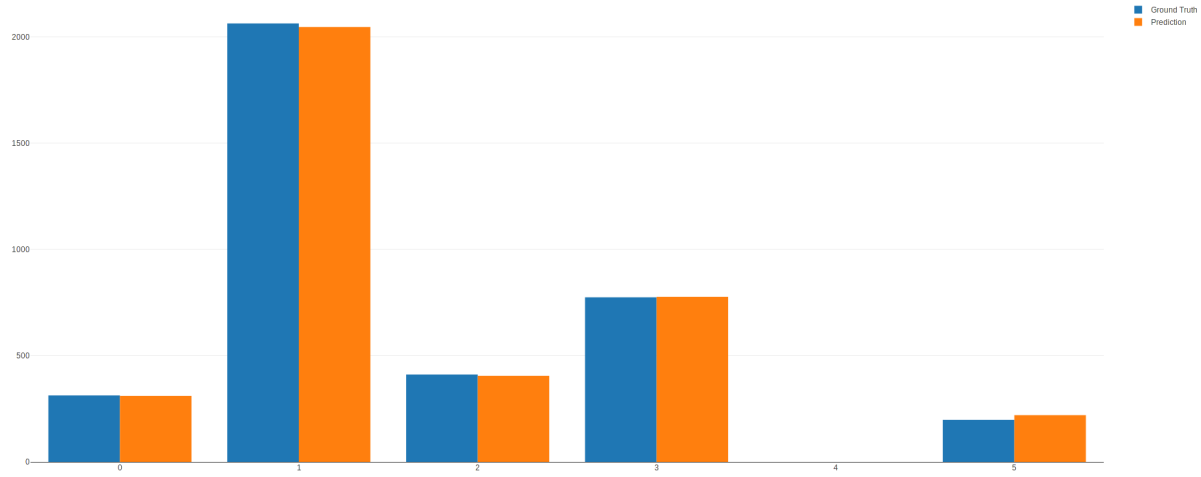


FIGURE B.6: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using CLA2.

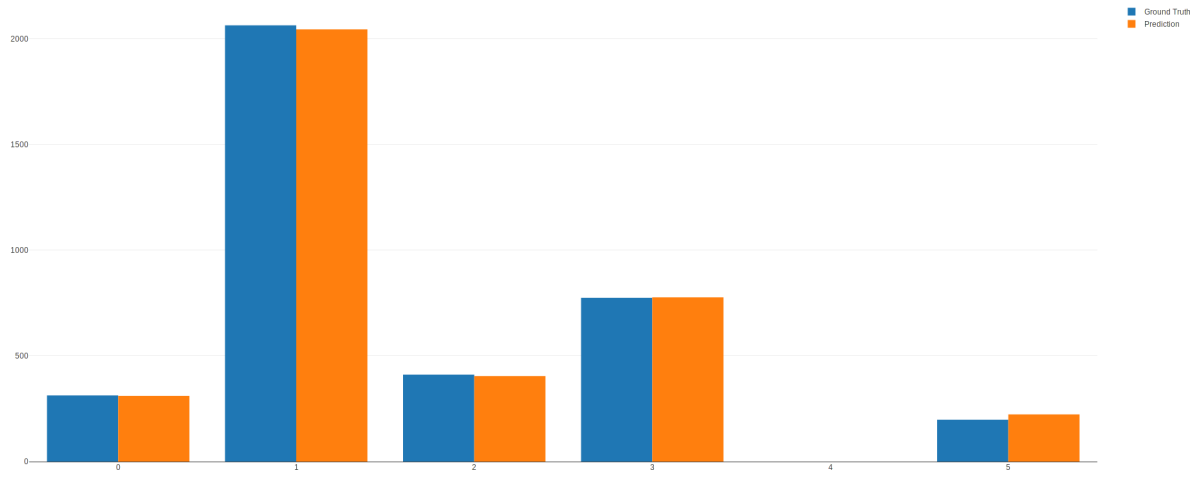


FIGURE B.7: The prediction accuracy with Ground Truth for the activities in the testing phase d1-1m-0tm using CLA3.

References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing* (pp. 304–307).: Springer.

Agrawal, P. & Franklin, S. (2014). Multi-layer cortical learning algorithms. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2014 IEEE Symposium on* (pp. 141–147).: IEEE.

Ahmad, S. & Hawkins, J. (2015). Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*.

Ahsan, U. & Bais, A. (2018). Distributed smart home architecture for data handling in smart grid. *Canadian Journal of Electrical and Computer Engineering*, 41(1), 17–27.

Alam, M. R., Reaz, M. B. I., & Ali, M. A. M. (2012). A review of smart homespast, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6), 1190–1203.

Alemdar, H., Ertan, H., Incel, O. D., & Ersoy, C. (2013a). Aras human activity datasets in multiple homes with multiple residents. In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare* (pp. 232–235).: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Alemdar, H., Ertan, H., Incel, O. D., & Ersoy, C. (2013b). Aras human activity datasets in multiple homes with multiple residents. In *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops* (pp. 232–235).: IEEE.

Alhafidh, B. M. H., Daood, A. I., & Allen, W. H. (2018). Comparison of classifiers for prediction of human actions in a smart home. In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on* (pp. 287–288).: IEEE.

Alpaydin, E. (2009). *Introduction to machine learning*. MIT press.

Alshammari, N., Alshammari, T., Sedky, M., Champion, J., & Bauer, C. (2017). Openshs: Open smart home simulator. *Sensors*, 17(5), 1003.

Alshammari, N. O. (2018). *Anomaly Detection Using Hierarchical Temporal Memory in Smart Homes*. PhD thesis, Staffordshire University.

Alshammari, T., Alshammari, N., Sedky, M., & Howard, C. (2018a). Evaluating machine learning techniques for activity classification in smart home environments. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 12(2), 48 – 54.

Alshammari, T., Alshammari, N., Sedky, M., & Howard, C. (2018b). Simadl: Simulated activities of daily living dataset. *Data*, 3(2), 11.

Amiribesheli, M., Benmansour, A., & Bouchachia, A. (2015). A review of smart homes in healthcare. *Journal of Ambient Intelligence and Humanized Computing*, 6(4), 495–517.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *ESANN*.

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine*, 5(4), 13–18.

Ariani, A., Redmond, S. J., Chang, D., & Lovell, N. H. (2013). Simulation of a smart home environment. In *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013 3rd International Conference on* (pp. 27–32).: IEEE.

Ashton, K. *et al.* (2009). That internet of things thing. *RFID journal*, 22(7), 97–114.

Assim, A., Reaz, M. B., Ibrahimy, M., Ismail, A., Choong, F., & Mohd-Yasin, F. (2006). An ai based self-moderated smart-home. *Informacije MIDEM*, 36(2), 91–94.

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15), 2787–2805.

Babakura, A., Sulaiman, M. N., Mustapha, N., & Perumal, T. (2014). Hmm-based decision model for smart home environment. *International Journal of Smart Home*, 8(1), 129–138.

Badica, C., Brezovan, M., & Badica, A. (2013). An overview of smart home environments: Architectures, technologies and applications. In *BCI (Local)* (pp.̃78).

Balasubramaniam, J., Krishnaa, C. G., & Zhu, F. (2015). Enhancement of classifiers in htm-cla using similarity evaluation methods. *Procedia Computer Science*, 60, 1516–1523.

Bandyopadhyay, D. & Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1), 49–69.

Bandyopadhyay, S., Sengupta, M., Maiti, S., & Dutta, S. (2011). Role of middleware for internet of things: A study. *International Journal of Computer Science and Engineering Survey*, 2(3), 94–105.

Barnes, N., Edwards, N., Rose, D., & Garner, P. (1998). Lifestyle monitoring-technology for supported independence. *Computing & Control Engineering Journal*, 9(4), 169–174.

Bartram, L., Rodgers, J., & Woodbury, R. (2011). Smart homes or smart occupants? supporting aware living in the home. In *IFIP Conference on Human-Computer Interaction* (pp. 52–64).: Springer.

Baum, L. E. & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6), 1554–1563.

Begg, R. & Hassan, R. (2006). Artificial neural networks in smart homes. In *Designing smart homes* (pp. 146–164). Springer.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2), 157–166.

Berlo, A., Bob, A., Jan, E., Klaus, F., Maik, H., & Charles, W. (1999). Design guidelines on smart homes: A cost 219bis guidebook. *Brussels, Belgium: European Commission*.

Bhati, A., Hansen, M., & Chan, C. M. (2017). Energy conservation through smart homes in a smart city: A lesson for singapore households. *Energy Policy*, 104, 230–239.

Bhattacharyya, D., Kim, T.-h., & Pal, S. (2010). A comparative study of wireless sensor networks and their routing protocols. *Sensors*, 10(12), 10506–10523.

Bouchard, K., Ajroud, A., Bouchard, B., & Bouzouane, A. (2010). Simact: a 3d open source smart home simulator for activity recognition. In *Advances in Computer Science and Information Technology* (pp. 524–533). Springer.

Bourobou, S. T. M. & Yoo, Y. (2015). User activity recognition in smart homes using pattern clustering applied to temporal ann algorithm. *Sensors*, 15(5), 11953–11971.

Brannen, J. (2005). Mixed methods research: A discussion paper.

Briere, D. & Hurley, P. (2011). *Smart homes for dummies*. John Wiley & Sons.

Brooke, J. *et al.* (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 4–7.

Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., & Zaccaria, R. (2013). Analysis of human behavior recognition algorithms based on acceleration data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (pp. 1602–1607).: IEEE.

Buchmayr, M., Kurschl, W., & Küng, J. (2011). A simulator for generating and visualizing sensor data for ambient intelligence environments. *Procedia Computer Science*, 5, 90–97.

Byrne, F. (2015). Encoding reality: Prediction-assisted cortical learning algorithm in hierarchical temporal memory. *arXiv preprint arXiv:1509.08255*.

Casale, P., Pujol, O., & Radeva, P. (2011). Human activity recognition from accelerometer data using a wearable device. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 289–296).: Springer.

CASAS (2009). Wsu casas datasets. `http://ailab.wsu.edu/casas/datasets/`. (accessed on 12 January 2017).

Catarci, T., Cincotti, F., De Leoni, M., Mecella, M., & Santucci, G. (2008). Smart homes for all: Collaborating services in a for-all architecture for domotics. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing* (pp. 56–69).: Springer.

Cavone, D., De Carolis, B., Ferilli, S., & Novielli, N. (2011). An agent-based approach for adapting the behavior of a smart home environment. In *WOA* (pp. 105–111).

Chan, M., Hariton, C., Ringeard, P., & Campo, E. (1995). Smart house automation system for the elderly and the disabled. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 2 (pp. 1586–1589).: IEEE.

Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., Millán, J. d. R., & Roggen, D. (2013). The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15), 2033–2042.

Chen, H., Lee, Y., Sun, G., Lee, H., Maxwell, T., & Giles, C. L. (1986). High order correlation model for associative memory. In *AIP Conference Proceedings*, volume 151 (pp. 86–99).: AIP.

Chen, Y.-H., Tsai, M.-J., Fu, L.-C., Chen, C.-H., Wu, C.-L., & Zeng, Y.-C. (2015). Monitoring elder's living activity using ambient and body sensor network in smart home. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on* (pp. 2962–2967).: IEEE.

Chen, Y.-K. (2012). Challenges and opportunities of internet of things. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific* (pp. 383–388).: IEEE.

Chollet, F. *et al.* (2015). Keras. `https://github.com/fchollet/keras`.

Classifiers NuPIC (2017). Cla classifier. `http://nupic.docs.numenta.org/0.6.0/classifiers.html`. (accessed on 15 November 2018).

Collet, Y. (2015). xxhash. `http://cyan4973.github.io/xxHash/`. (accessed on 05 February 2017).

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 1–8).: Association for Computational Linguistics.

Cook, D., Schmitter-Edgecombe, M., Crandall, A., Sanders, C., & Thomas, B. (2009). Collecting and disseminating smart home sensor data in the casas project. In *Proceedings of the CHI workshop on developing shared home behavior datasets to advance HCI and ubiquitous computing research* (pp. 1–7).

Cook, D. J. (2009). Multi-agent smart environments. *Journal of Ambient Intelligence and Smart Environments*, 1(1), 51–55.

Cook, D. J., Crandall, A. S., Thomas, B. L., & Krishnan, N. C. (2013). Casas: A smart home in a box. *Computer*, 46(7).

Cook, D. J. & Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5), 480.

Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. (2003). Mavhome: An agent-based smart home. In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on* (pp. 521–524).: IEEE.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.

Cui, Y., Ahmad, S., & Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, 28(11), 2474–2504.

Cui, Y., Ahmad, S., & Hawkins, J. (2017). The htm spatial poolera neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, 11.

Darwish, A. & Hassanien, A. E. (2011). Wearable and implantable wireless sensor network solutions for healthcare monitoring. *Sensors*, 11(6), 5561–5595.

Das, S. K., Cook, D. J., Battacharya, A., Heierman, E. O., & Lin, T.-Y. (2002). The role of prediction algorithms in the mavhome smart home architecture. *IEEE Wireless Communications*, 9(6), 77–84.

De Ruyter, B., Aarts, E., Markopoulos, P., & Ijsselsteijn, W. (2005). Ambient intelligence research in homelab: Engineering the user experience. In *Ambient Intelligence* (pp. 49–61). Springer.

Devineni, A. (2015). How our brains learn. `http://www.brains-explained.com/how-our-brains-learn`. (accessed on 11 October 2018).

Dixit, A. & Naik, A. (2014). Use of prediction algorithms in smart homes. *International Journal of Machine Learning and Computing*, 4(2), 157.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87.

Duckham, M. & Bennett, R. (2009). Ambient spatial intelligence. In *BMI Book* (pp. 319–335).

Elkhodr, M., Shahrestani, S., & Cheung, H. (2016). The internet of things: new interoperability, management and security challenges. *arXiv preprint arXiv:1604.04824*.

Eluyode, O. & Akomolafe, D. T. (2013). Comparative study of biological and artificial neural networks. *European Journal of Applied Engineering and Scientific Research*, 2(1), 36–46.

Emi, I. A. & Stankovic, J. A. (2015). Sarrima: smart adl recognizer and resident identifier in multi-resident accommodations. In *Proceedings of the conference on Wireless Health* (pp.4̃).: ACM.

Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011), 1–11.

Fallas-Moya, F. (2015). Object tracking based on hierarchical temporal memory classification.

Fang, H. & He, L. (2012). Bp neural network for human activity recognition in smart home. In *Computer Science & Service System (CSSS), 2012 International Conference on* (pp. 1034–1037).: IEEE.

Farhadi, F. (2017). *Learning Activation Functions in Deep Neural Networks*. PhD thesis, École Polytechnique de Montréal.

Fatima, I., Fahim, M., Lee, Y.-K., & Lee, S. (2012). Effects of smart home dataset characteristics on classifiers performance for human activity recognition. In *Computer Science and its Applications* (pp. 271–281). Springer.

Fleury, A., Vacher, M., & Noury, N. (2010). 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.

Fortino, G., Guerrieri, A., & Russo, W. (2012). Agent-oriented smart objects development. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* (pp. 907–912).: IEEE.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119–139.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2), 127.

Fu, Q., Li, P., Chen, C., Qi, L., Lu, Y., & Yu, C. (2011). A configurable context-aware simulator for smart home systems. In *6th International Conference on Pervasive Computing and Applications (ICPCA)* (pp. 39–44).: IEEE.

Gartner (2017). 8.4 billion connected things will be in use in 2017, up 31 percent from 2016. `http://www.gartner.com/newsroom/id/3598917`. (accessed on 26 February 2017).

George, D. (2008). *How the brain might work: A hierarchical and temporal model for learning and recognition*. Stanford University.

Gomez, C. & Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6).

Gopalratnam, K. & Cook, D. J. (2004). Active lezi: An incremental parsing algorithm for sequential prediction. *International Journal on Artificial Intelligence Tools*, 13(04), 917–929.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7), 1645–1660.

Gunn, S. R. *et al.* (1998). Support vector machines for classification and regression. *ISIS technical report*, 14(1), 5–16.

Haller, S., Karnouskos, S., & Schroth, C. (2009). The internet of things in an enterprise context, future internet—fis 2008: First future internet symposium, fis 2008 vienna, austria, september 29-30, 2008 revised selected papers.

Harle, R. K. & Hopper, A. (2008). The potential for location-aware power management. In *Proceedings of the 10th international conference on Ubiquitous computing* (pp. 302–311).: ACM.

Hawkins, J. & Ahmad, S. (2016). Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10, 23.

Hawkins, J., Ahmad, S., & Dubinsky, D. (2011). Hierarchical temporal memory including htm cortical learning algorithms, 0.2. numenta. *Techical report, Numenta, Inc, Palto Alto http://www. numenta. com/htmoverview/education/HTM_CorticalLearningAlgorithms. pdf*.

Hawkins, J., Ahmad, S., Purdy, S., & Lavin, A. (2016). Biological and machine intelligence (bami). *Initial online release 0.4*.

Hawkins, J. & Blakeslee, S. (2004a). *On intelligence*. New York, NY: St. Martin's Press.

Hawkins, J. & Blakeslee, S. (2004b). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan.

Hawkins, J. & Blakeslee, S. (2007). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan.

Haykin, S. S., Haykin, S. S., Haykin, S. S., & Haykin, S. S. (2009). *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River.

Heaton, J. (2008). *Introduction to neural networks with Java*. Heaton Research, Inc.

Hebb, D. O. (1949). The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology.*, (pp. 62–78).

Helal, A., Cho, K., Lee, W., Sung, Y., Lee, J., & Kim, E. (2012). 3d modeling and simulation of human activities in smart spaces. In *Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on* (pp. 112–119).: IEEE.

Helal, S., Lee, J. W., Hossain, S., Kim, E., Hagras, H., & Cook, D. (2011). Persim-simulator for human activities in pervasive spaces. In *7th International Conference on Intelligent Environments (IE)* (pp. 192–199).: IEEE.

Held, R., Freedman, S., & Harris, C. (1996). Activation of the primary visual cortex by braille reading in blind subjects. *Nature*, 380, 11.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hussein, A., Adda, M., Atieh, M., & Fahs, W. (2014). Smart home design for disabled people based on neural networks. *Procedia Computer Science*, 37, 117–126.

Issarny, V., Caporuscio, M., & Georgantas, N. (2007). A perspective on the future of middleware-based software engineering. In *Future of Software Engineering, 2007. FOSE'07* (pp. 244–258).: IEEE.

Jahn, M., Jentsch, M., Prause, C. R., Pramudianto, F., Al-Akkad, A., & Reiners, R. (2010). The energy aware smart home. In *Future Information Technology (FutureTech), 2010 5th International Conference on* (pp. 1–8).: IEEE.

Jeff Hawkins (2014). Principles of hierarchical temporal memory - foundations of machine intelligence. `https://www.slideshare.net/numenta/2014-10-17-numenta-workshop`. (accessed on 17 March 2017).

Jih, W.-r., Hsu, J. Y.-j., Wu, C.-L., Liao, C.-F., & Cheng, S.-y. (2006). A multi-agent service framework for context-aware elder care. In *In AAMAS*.

Johnston, J. (2008). *The allure of machinic life: cybernetics, artificial life, and the new AI*. MIT Press.

Juels, A. (2006). Rfid security and privacy: A research survey. *IEEE journal on selected areas in communications*, 24(2), 381–394.

Kawsar, F. (2009). A document-based framework for user centric smart object systems. *PhD in Computer Science, Waseda University, Japan*.

Kawsar, F. & Nakajima, T. (2009). A document centric framework for building distributed smart object systems. In *Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC'09. IEEE International Symposium on* (pp. 71–79).: IEEE.

Kientz, J. A., Patel, S. N., Jones, B., Price, E., Mynatt, E. D., & Abowd, G. D. (2008). The georgia tech aware home. In *CHI'08 extended abstracts on Human factors in computing systems* (pp. 3675–3680).: ACM.

Kneller, A. & Thornton, J. (2015). Distal dendrite feedback in hierarchical temporal memory. In *Neural Networks (IJCNN), 2015 International Joint Conference on* (pp. 1–8).: IEEE.

Kormányos, B. & Pataki, B. (2013). Multilevel simulation of daily activities: Why and how? In *2013 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)* (pp. 1–6).: IEEE.

Lago, P., Lang, F., Roncancio, C., Jiménez-Guarín, C., Mateescu, R., & Bonnefond, N. (2017). The contextact@ a4h real-life dataset of daily-living activities. In *International and Interdisciplinary Conference on Modeling and Using Context* (pp. 175–188).: Springer.

Lavesson, N. & Davidsson, P. (2006). Quantifying the impact of learning algorithm parameter tuning. In *AAAI*, volume 6 (pp. 395–400).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

LeCun, Y. & Ranzato, M. (2013). Deep learning tutorial. In *Tutorials in International Conference on Machine Learning (ICML13)* (pp. 1–29).: Citeseer.

Lee, J. W., Cho, S., Liu, S., Cho, K., & Helal, S. (2015). Persim 3D : Context-Driven Simulation and Modeling of Human Activities in Smart Spaces. *IEEE Transactions on Automation Science and Engineering*, 12(4), 1243–1256.

Lee, R. & Rajabi, M. (2014). Assessing nupic and cla in a machine learning context using nasa aviation datasets.

Leibold, C. & Kempter, R. (2006). Memory capacity for sequences in a recurrent network with biological constraints. *Neural computation*, 18(4), 904–941.

Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., & Zhang, S. (1999a). A multi-agent system for intelligent environment control. In *Proceedings of the Third International Conference on Autonomous Agents* (pp. 291–298).

Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., & Zhang, S. X. (1999b). The intelligent home testbed. *environment*, 2, 15.

Liu, X., Ramirez, S., Pang, P. T., Puryear, C. B., Govindarajan, A., Deisseroth, K., & Tonegawa, S. (2012). Optogenetic stimulation of a hippocampal engram activates fear memory recall. *Nature*, 484(7394), 381.

Lobaccaro, G., Carlucci, S., & Löfström, E. (2016). A review of systems and technologies for smart homes and smart grids. *Energies*, 9(5), 348.

Logan, B. & Healey, J. (2006). Sensors to detect the activities of daily living. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE* (pp. 5362–5365).: IEEE.

Lundström, J., Synnott, J., Järpe, E., & Nugent, C. D. (2015). Smart home simulation using avatar control and probabilistic sampling. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on* (pp. 336–341).: IEEE.

Lutolf, R. (1992). Smart home concept and the integration of energy meters into a home based system. In *Metering Apparatus and Tariffs for Electricity Supply, 1992., Seventh International Conference on* (pp. 277–278).: IET.

Manogaran, G. & Lopez, D. (2017). Health data analytics using scalable logistic regression with stochastic gradient descent. *International Journal of Advanced Intelligence Paradigms*, 8(2).

Marsland, S. (2011). *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.

Mattsson, O. (2011). Fruit recognition by hierarchical temporal memory.

McCarthy, J. (2007). What is artificial intelligence?

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4), 12.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115–133.

McDonald, H., Nugent, C., Hallberg, J., Finlay, D., Moore, G., & Synnes, K. (2013). The homeml suite: shareable datasets for smart home environments. *Health and Technology*, 3(2), 177–193.

Meditskos, G., Dasiopoulou, S., & Kompatsiaris, I. (2016). Metaq: A knowledge-driven framework for context-aware activity recognition combining sparql and owl 2 activity patterns. *Pervasive and Mobile Computing*, 25, 104–124.

Micucci, D., Mobilio, M., & Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10), 1101.

Minor, B., Doppa, J. R., & Cook, D. J. (2015). Data-driven activity prediction: Algorithms, evaluation methodology, and applications. In *Proceedings of the 21th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 805–814).: ACM.

Minsky, M. & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7), 1497–1516.

Mozer, M., Dodier, R., Miller, D., Anderson, M., Anderson, J., Bertini, D., Bronder, M., Colagrosso, M., Cruickshank, R., Daugherty, B., *et al.* (2005). The adaptive house. In *IEE Seminar Digests*, volume 11059 (pp. v1–39).: IET.

Mozer, M. C. (1998). The neural network house: An environment hat adapts to its inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments*, volume 58.

Neagle, C. (2014). A guide to the confusing internet of things standards world. *networkWorld, Network World*.

Ni, Q., García Hernando, A. B., & de la Cruz, I. P. (2015). The elderlys independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development. *Sensors*, 15(5), 11312–11362.

Numenta CLA Classifier (2017). Cla classifier. `https://discourse.numenta.org/t/cla-classifier/2143`. (accessed on 15 November 2018).

Numenta KNN Classifier (2017). Knn classifier. `http://nupic.docs.numenta.org/1.0.0/api/algorithms/classifiers.html#knn-classifier`. (accessed on 15 November 2018).

Ordónez, F. J., de Toledo, P., & Sanchis, A. (2013). Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13(5), 5460–5477.

Ordóñez, F. J. & Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 115.

Otahal, M. & Stepankova, O. (2014). Anomaly detection with cortical learning algorithm for smart homes. *SMART HOMES 2014*, (pp.2̃4).

Pauwels, E. J., Salah, A. A., & Tavenard, R. (2007). Sensor networks for ambient intelligence. In *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on* (pp. 13–16).: IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,

D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Perera, C., Liu, C. H., Jayawardena, S., & Chen, M. (2014a). A survey on internet of things from industrial market perspective. *IEEE Access*, 2, 1660–1679.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014b). Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1), 414–454.

Pipattanasomporn, M., Kuzlu, M., & Rahman, S. (2012). An algorithm for intelligent home energy management and demand response analysis. *IEEE Transactions on Smart Grid*, 3(4), 2166–2173.

Pirbhulal, S., Zhang, H., E Alahi, M., Ghayvat, H., Mukhopadhyay, S., Zhang, Y.-T., & Wu, W. (2017). A novel secure iot-based smart home automation system using a wireless sensor network. *Sensors*, 17(1), 69.

Pirsiavash, H. & Ramanan, D. (2012). Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 2847–2854).: IEEE.

PlaceLab (2005). Placelab datasets. `http://web.mit.edu/cron/group/house_n/data/PlaceLab/PlaceLab.htm`. (accessed on 05 February 2017).

Poland, M. P., Nugent, C. D., Wang, H., & Chen, L. (2009). Development of a smart home simulator for use as a heuristic tool for management of sensor distribution. *Technology and Health Care*, 17(3), 171–182.

Poo, M.-m., Pignatelli, M., Ryan, T. J., Tonegawa, S., Bonhoeffer, T., Martin, K. C., Rudenko, A., Tsai, L.-H., Tsien, R. W., Fishell, G., *et al.* (2016). What is memory? the present state of the engram. *BMC biology*, 14(1), 40.

Price, R. W. (2011). *Hierarchical temporal memory cortical learning algorithm for pattern recognition on multi-core architectures*. Portland State University.

Prossegger, M. & Bouchachia, A. (2014). Multi-resident activity recognition using incremental decision trees. In *Adaptive and Intelligent Systems* (pp. 182–191). Springer.

Purdy, S. (2016). Encoding data for htm systems. *arXiv preprint arXiv:1602.05925*.

Quesada, F. J., Moya, F., Medina, J., Martínez, L., Nugent, C., & Espinilla, M. (2015). Generation of a partitioned dataset with single, interleave and multioccupancy daily living activities. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 60–71).: Springer.

Rao, R. P. & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1), 79.

Rashidi, P. & Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *IEEE journal of biomedical and health informatics*, 17(3), 579–590.

Robert, H.-N. (1989). A basic introduction to neural networks.

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

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

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.

Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533.

Russell, S. J. & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Saad al sumaiti, A., Ahmed, M. H., & Salama, M. M. (2014). Smart home activities: A literature review. *Electric Power Components and Systems*, 42(3-4), 294–305.

Sanchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E., *et al.* (2014). Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61, 217–238.

Saponara, S. & Bacchillone, T. (2012). Network architecture, security issues, and hardware implementation of a home area network for smart grid. *Journal of Computer Networks and Communications*, 2012.

Sarma, A. C. & Girão, J. (2009). Identities in the future internet of things. *Wireless personal communications*, 49(3), 353–363.

Satpathy, L. (2006). *Smart Housing: Technology to Aid Aging in Place: New Opportunities and Challenges*. Mississippi State University.

SBCI, U. (2007). Buildings and climate change: status, challenges and opportunities. *United Nations Environment Programme (UNEP), Paris*.

Schilit, B. N. & Theimer, M. M. (1994). Disseminating active map information to mobile hosts. *IEEE network*, 8(5), 22–32.

Shah, B., Pattanayak, R. D., Sagar, R., *et al.* (2014). The study of patient henry molaison and what it taught us over past 50 years: Contributions to neuroscience. *Journal of Mental Health and Human Behaviour*, 19(2), 91.

Sharma, J., Angelucci, A., & Sur, M. (2000). Induction of visual orientation modules in auditory cortex. *Nature*, 404(6780), 841.

Siganos, D. & Stergiou, C. (1996). Neural networks, the human brain, and learning. *Imperial College London: Surveys and Presentations in Information Systems Engineering*.

Singla, G., Cook, D. J., & Schmitter-Edgecombe, M. (2009). Tracking activities in complex settings using smart environment technologies. *International journal of bioscience, psychiatry, and technology (IJBSPT)*, 1(1), 25.

Singla, G., Cook, D. J., & Schmitter-Edgecombe, M. (2010). Recognizing independent and joint activities among multiple residents in smart environments. *Journal of ambient intelligence and humanized computing*, 1(1), 57–63.

Škoviera, R. & Bajla, I. (2013). Image classification based on hierarchical temporal memory and color features. *Slovak Academy of Sciences, MEASUREMENT*.

Skubic, M., Alexander, G., Popescu, M., Rantz, M., & Keller, J. (2009a). A smart home application to eldercare: Current status and lessons learned. *Technology and Health Care*, 17(3), 183–201.

Skubic, M., Alexander, G., Popescu, M., Rantz, M., & Keller, J. (2009b). A smart home application to eldercare: Current status and lessons learned. *Technology and Health Care*, 17(3), 183–201.

Speer, B., Miller, M., Schaffer, W., Gueran, L., Reuter, A., Jang, B., & Widegren, K. (2015). *Role of smart grids in integrating renewable energy*. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Srivastava, A. K. & Saxena, S. (2006). Internet of things and its big player.

Subutai, A. (2015). Getting predictions out of htm (cla classifier).

Synnott, J., Chen, L., Nugent, C., & Moore, G. (2014). The creation of simulated activity datasets using a graphical intelligent environment simulation tool. (pp. 4143–4146).

Synnott, J., Nugent, C., & Jeffers, P. (2015). Simulation of smart home activity datasets. *Sensors*, 15(6), 14162–14179.

Taleb, T., Bottazzi, D., Guizani, M., & Nait-Charif, H. (2009). Angelah: a framework for assisting elders at home. *IEEE Journal on Selected Areas in Communications*, 27(4).

Tapia, E. M., Intille, S. S., & Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing* (pp. 158–175).: Springer.

Torunski, E., Othman, R., Orozco, M., & El Saddik, A. (2012). A review of smart environments for energy savings. *Procedia Computer Science*, 10, 205–214.

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test* (pp. 23–65). Springer.

Vainio, A.-M., Valtonen, M., & Vanhala, J. (2008). Proactive fuzzy control and adaptation methods for smart homes. *IEEE Intelligent Systems*, (2), 42–49.

Van Kasteren, T., Englebienne, G., & Kröse, B. J. (2010). Transferring knowledge of activity recognition across sensor networks. In *International Conference on Pervasive Computing* (pp. 283–300).: Springer.

Van Kasteren, T., Noulas, A., Englebienne, G., & Kröse, B. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing* (pp. 1–9).: ACM.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4), 395–416.

Webber, F. D. S. (2015). Semantic folding theory and its application in semantic fingerprinting. *arXiv preprint arXiv:1511.08855*.

Weber, R. H. (2010). Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1), 23–30.

Weiss, M., Mattern, F., Graml, T., Staake, T., & Fleisch, E. (2009). Handy feedback: Connecting smart meters with mobile phones. In *Proceedings of the 8th international conference on mobile and ubiquitous multimedia* (pp.1̃5).: ACM.

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., & Borriello, G. (2009). Building the internet of things using rfid: the rfid ecosystem experience. *IEEE Internet computing*, 13(3).

Williams, C. (2007). Research methods, journal of business & economic research–march 2007 volume 5, number 3.

Winkler, B. (2002). *An implementation of an ultrasonic indoor tracking system supporting the OSGI architecture of the ICTA lab*. PhD thesis, University of Florida.

Yang, Q., Tan, K.-H., Culbertson, B., & Apostolopoulos, J. (2010). Fusion of active and passive sensors for fast 3d capture. In *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on* (pp. 69–74).: IEEE.

Youngblood, G. M., Cook, D. J., & Holder, L. B. (2005a). A learning architecture for automating the intelligent environment. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20 (pp. 1576).: Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Youngblood, M., Cook, D. J., & Holder, L. B. (2005b). Seamlessly engineering a smart environment. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 1 (pp. 548–553).: IEEE.

Yuwei, C. (2017). Sdr classifier. `https://github.com/ywcui1990`. (accessed on 15 November 2018).

Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., & Troster, G. (2008). Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. *Lecture Notes in Computer Science*, 4913, 17.

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., & Zhang, J. (2014). Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on* (pp. 197–205).: IEEE.

Zhang, X., Zhang, J., & Zhong, J. (2017). Skill learning for intelligent robot by perception-action integration: A view from hierarchical temporal memory. *Complexity*, 2017.

Zhu, L., Chen, Y., Ye, X., & Yuille, A. (2008). Structure-perceptron learning of a hierarchical log-linear model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1–8).: IEEE.

Zurada, J. M. (1992). *Introduction to artificial neural systems*, volume 8. West St. Paul.