Anomaly Detection Using Hierarchical Temporal Memory in Smart Homes

Nasser Owaid Alshammari

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

March 2018

List of Publications

REFEREED JOURNAL PUBLICATIONS

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

REFEREED CONFERENCE PUBLICATIONS

Exploring the Adoption of Physical Security Controls in Smartphones **August,2015**
International Conference on Human Aspects of Information Security, Privacy, and Trust

SOFTWARE

OpenSHS `http://www.openshs.org`
A new open-source, 3D, cross-platform smart home simulator that follows a hybrid approach for generating representative datasets for smart home research.

Acknowledgements

I would like to thank my supervisors Dr. Mohamed Sedky, Dr. Justin Champion and Dr. Carolin Bauer for their dedicated supervision during my Ph.D study. Their endless help, kindness, patience, and thoughtful considerations are greatly appreciated.

I would like to thank my family for their unlimited support which made my Ph.D journey possible.

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

Thanks also go to my fellow researchers and colleagues for the good time we had in Staffordshire University.

Contents

Abstract

This work focuses on unsupervised biologically-inspired machine learning techniques and algorithms that can detect anomalies. Specifically, the aim is to investigate the applicability of the Hierarchical Temporal Memory (HTM) theory in detecting anomalies in the smart home domain. The HTM theory proposes a model for the neurons that is more faithful to the actual neurons than their usual counterparts in Artificial Neural Networks (ANN) based on the current Neuroscience understanding. The HTM theory has several algorithmic implementations, the most prominent one is the Cortical Learning Algorithm (CLA). The CLA model typically consists of three main regions: the encoder, the spatial pooler and the temporal memory. Studying the performance of the CLA in the smart home domain revealed an issue with the standard encoders and high-dimensional datasets. In this domain, it is typical to have high-dimensional feature space representing the collection of smart devices. The standard CLA encoders are more suitable for low-dimensional datasets and there are encoders for categorical and scalar data types.

A novel Hash Indexed Sparse Distributed Representation (HI-SDR) encoder was proposed and developed, to overcome the high-dimensionality issue. The HI-SDR encoder creates unique representation of the data which allows the rest of the CLA regions to learn from. The standard approach when creating HTM models to work with datasets with many features is to concatenate the output of each encoder. This work concludes that the standard encoders produced representations for the input during every timestep that were similar and less distinguishable for the HTM model. This output similarity confuses the HTM model and makes it hard to discern meaningful representations. The proposed novel encoder manages to capture the required properties in terms of sparsity and representations.

To investigate and validate the performance of a proposed machine learning technique, there has to be a representative dataset. In the smart home literature, there exists many real-world smart home datasets that allow the researchers to validate their models. However, most of the existing datasets are created for classification and recognition of Activities of Daily Living (ADL). The lack of datasets for anomaly detection applications in the domain of smart homes required the development of a simulation tool. OpenSHS (Open Smart Home Simulator) was developed as an open-source, 3D and cross-platform smart home simulator that offers a novel hybrid approach to dataset generation. The tool allows the researchers to design a smart home and populate it with the needed smart devices. Then, the participants can use the designed smart home and simulate their habits and patterns.

Anomaly detection in the smart home domain is highly contextual and dependent on the inhabitant's activities. One inhabitant's anomaly could be the norm for another,

therefore the definition of anomalies is a complex consideration. Using OpenSHS, seven participants were invited to generated forty-two datasets of their activities. Moreover, each participant defined his/her own anomalous pattern that he/she would like the model to detect. Thus, the resulting datasets are annotated with contextual anomalies specific to each participant.

The proposed encoder has been evaluated and compared against the standard CLA encoders and several state-of-the-art unsupervised anomaly detection algorithms, using Numenta Anomaly Benchmark (NAB). The HI-SDR encoder scored 81.9% accuracy, on the forty-two datasets, with 17.8% increase in accuracy compared to the k-NN algorithm and 47.5% increase over the standard CLA encoders. Using the Principal Component Analysis (PCA) algorithm as a preprocessing step proved to be beneficial to some of the tested algorithms. The k-NN algorithm scored 39.9% accuracy without PCA and scored 64.1% accuracy with PCA. Similarly, the Histogram Based Outlier Score (HBOS) algorithm scored 28.5% accuracy without PCA and 61.9% with PCA.

The HTM-based models empirically showed good potential and exceeded in performance several algorithms, even without the HI-SDR encoder. However, the HTM-based models still lack an optimisation algorithm for its parameters when performing anomaly detection.

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
aLOCI	**a**pproximate **LO**cal **C**orrelation **I**ntegral
CBLOF	**C**luster-**B**ased **L**ocal **O**utlier **F**actor
CMGOS	**C**lustering-Based **M**ultivariate **G**aussian **O**utlier **S**core
CNN	**C**onvolutional **N**eural **N**etworks
CLA	**C**ortical **L**earning **A**lgorithm
COF	**C**onnectivity-Based **O**utlier **F**actor
HI-SDR	**H**ash **I**ndexed **S**parse **D**istributed **R**epresentations
HBOS	**H**istogram-**B**ased **O**utlier **S**core
HTM	**H**ierarchical **T**emporal **M**emory
IoT	**I**nternet **o**f **T**hings
KNN	**K** **N**earest **N**eighbour
LDCOF	**L**ocal **D**ensity **C**luster-Based **L**ocal **O**utlier **F**actor
LOF	**L**ocal **O**utlier **F**actor
LOCI	**LO**cal **C**orrelation **I**ntegral
LoOP	**Lo**cal **O**utlier **F**actor
NAB	**N**umenta **A**nomaly **B**enchmark
NuPIC	**Nu**menta **P**latform for **I**ntelligent **C**omputing
OpenSHS	**O**pen **S**mart **H**ome **S**imulator
PCA	**P**rincipal **C**omponent **A**nalysis
PSO	**P**article **S**warm **O**ptimisation
RFID	**R**adio **F**requency **ID**entification
SDR	**S**parse **D**istributed **R**epresentations
SP	**S**patial **P**ooler
TM	**T**emporal **M**emory

“The key to artificial intelligence has always been the representation.”

Jeff Hawkins

Chapter 1

Introduction

The Internet of Things (IoT) paradigm imposes several challenges on today's technologies. With the increasing demands of the IoT, a revolution on the hardware and software levels is required. Taking into account the huge amount of data generated by the smart devices in the IoT vision, anomaly detection becomes a valuable tool in managing and controlling that flood of data. 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 will reach $273 billion by the end of 2017.

Today's smart homes are equipped with many sensors and actuators that generate a stream of spatio-temporal data representing the state of the home and the inhabitants. Recent advancements in hardware and communication protocols allowed the vision of the IoT to become a reality. Under the IoT paradigm, many of the smart home devices gain the ability to connect to the public Internet and perform complex tasks. This necessitates the existence of a management and controlling mechanism for these smart devices. The challenge is how to integrate and evaluate the collected data to recognise abnormal activities, such as leaving the main door left open overnight. The smart home literature studies many areas and applications with machine learning playing an important role in these areas and can progress the capabilities of the smart home middlewares. One of the applications that machine learning models offer is the ability to detect anomalous activities in a dataset. Anomaly detection is essential for many applications in the smart home such as privacy, security, and elderly care. Current machine learning techniques fall short of reaching the full potential of anomaly detection. However, we know that the human brain is more than capable of detecting anomalous patterns. Thus,

machine learning techniques that simulate what actually happens in the human brain will help pushing the research towards smarter anomaly detection.

One of the recent advancements and efforts to understand how the human brain performs some of its cognitive operations is the Hierarchical Temporal Memory (HTM) theory (Hawkins & Ahmad, 2016). Backed by recent findings in Neuroscience, the HTM theory suggests an overarching view of how the neocortex learns from sensory input. The Cortical Learning Algorithm (CLA) (Ahmad *et al.*, 2017) is an algorithm influenced by the principals of the HTM theory that can perform anomaly detection in streams of data without the need for supervision.

Detecting anomalies is one of the important challenges facing smart home research. Anomaly detection in the smart home domain can serve many purposes in many areas such as detecting abnormal power consumption, detecting suspicious activities and monitoring the elderly especially those who live independently. Anomaly detection can help in providing intelligent services that are context-aware of the smart home user. However, one of the challenges facing any machine learning model is how to model the context and daily patterns and behaviour of the smart home user. This is challenging due to how random and noisy these patterns are. Another challenge in the smart home research field is having a good dataset that captures the different interactions that happen between the inhabitants and their environment.

This research could be used to implement an anomaly detection interface that can be used in an IoT middleware. Let us consider an example of a smart home user that goes to work on weekdays, and he/she always picks up his/her car keys before leaving the house. In the case where he/she forgets to take the car keys, the anomaly detection layer should detect this as an anomaly, and it is the middleware's responsibility to provide the appropriate action.

1.1 Aim

The aim of this research is to investigate the use and application of the HTM theory to propose an intelligent and adaptive anomaly detection technique that can be used as a part of a middleware layer for the smart home user in the IoT era.

1.2 Objectives

The objectives of this research are to:

1. Conduct a literature review of existing IoT middlewares and to critically evaluate their strengths and weaknesses,
2. Conduct a literature review of anomaly detection research in the context of smart homes,
3. Investigate the use of HTM as an adaptive anomaly detection method for smart homes,
4. Propose a design and an implementation of a simulation and testing environment for smart homes under the IoT vision,
5. Build a dataset using the simulation environment that simulates the smart home user's daily patterns,
6. Study different context modelling methods in the IoT paradigm,
7. Propose a novel semantic representation to facilitate context modelling in an IoT smart home environment,
8. Evaluate the proposed context modelling representation to detect anomalies in a smart home setting against state-of-the-art machine learning algorithms.

1.3 Scope of the Investigation

The IoT vision requires many revolutions on many fronts. It will be impossible to study and address all the issues that are facing the full adoption of the IoT vision. Therefore, this research will make some assumptions and will position itself in a well-defined place to address some of the research gaps.

The smart home domain is one of the realisations of the IoT vision. The research at hand, focuses on detecting anomalous patterns of the smart home inhabitants via machine learning models. However, there are many aspects and issues that surround the actual implementation of such techniques and it would be impossible to address all of these issues. Therefore, the following points are the main assumptions that this research assumes:

- **Reliable communication:** The research will not address low-level issues regarding the transportation of packets in the network. The assumption here is that all the devices and the network are capable of providing a reliable networking and transportation. Also, the research will not address the different protocols or means of communication, like communicating over WiFi or Bluetooth, etc.

- **Reliable addressing:** How the devices will be identified in the network and what addressing scheme will be used is outside of the scope of the research.
- **Reliable storage:** Storage and retrieval of data for all components are assumed to be reliable and capable of good performance.
- **Reliable processing:** The smart devices are assumed to have enough processing power to allow them to perform the desired tasks.

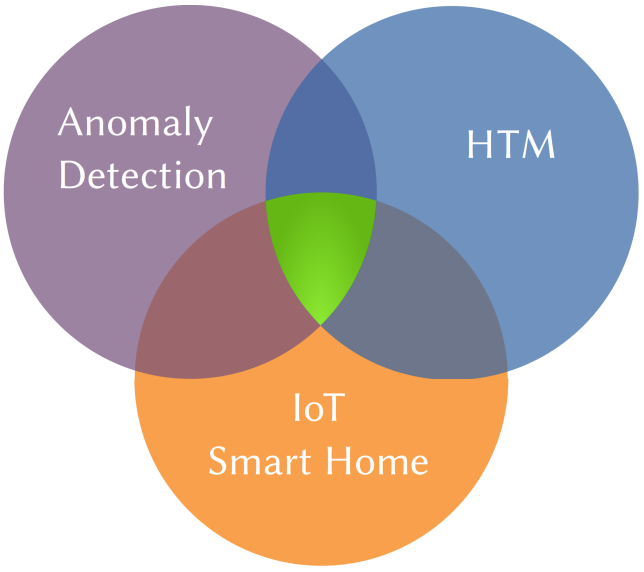


FIGURE 1.1: The scope of the investigation.

Figure 1.1 illustrates the three broad fields that this research touches on and the interaction of them which are the focus and scope of this work. In Chapter 2, the IoT issues and applications in general will be presented. However, the key part from that domain is the smart home as one of the IoT prominent applications.

Anomaly detection as a field of study is a well-established discipline and many mathematical and statistical research efforts were dedicated to investigate the various aspects of detecting outliers out of a collection of data. However, due to the influx of data that the state-of-the-art techniques provide, the interest of this field is rising. The nature of the data and the requirement for fast anomaly detection impose several challenges on any proposed technique.

The data in a smart home setting is streaming and spatio-temporal. Moreover, the inhabitants' activities are very subjective and differs from on inhabitant to another. This

subjectivity makes it hard to decide what an anomaly actually is. Therefore, this research will focus on unsupervised anomaly detection techniques because these techniques do not require the existence of labels to train from. The unsupervised techniques will learn from the data and predict if a certain activity is an anomaly or not based on the inhabitant's own daily patterns.

The HTM theory, which will be presented in details in Chapter 3, takes the time as an important factor and as one of its principal components in the learning process. Moreover, the theory proposes a flexible representation of the data that is inspired by the recent advancements in Neuroscience. Many implementations of the theory exists in the literature. These implementations provide classification, prediction, and anomaly detection services. However, the investigation of the application of this theory in a smart home setting to detect anomalies has not been studied in the literature.

1.4 Contributions to Knowledge

The novel contributions of this work are summarised as:

- The primary contribution is a novel HTM encoder suitable for detecting anomalies in smart homes. The standard HTM scalar encoders and categorical encoders did not produce good results when encoding high-dimensional datasets. The novel encoder, the Hash-Indexed Sparse Distributed Representation encoder (HI-SDR) (see Chapter 5), is able to capture the required properties for good HTM encoders (see Section 5.3.1) and encode high-dimensional dataset into a representation that allows the rest of the HTM model components to learn and detect anomalies with better accuracy.
- A secondary contribution of this work is OpenSHS (see Chapter 4). OpenSHS is A new open-source, 3D, cross-platform smart home simulator that follows a hybrid approach for generating representative datasets for smart home research,
- Another secondary contribution is the creation of forty-two smart home datasets created by seven participants using OpenSHS specifically for anomaly detection problems. One of the significant contributions of these datasets is that the problematic definition of anomalies in the context of smart homes has been dealt with in a user-centric way. Each participant, performed a series of anomalous events according to their habits and behaviour. Thus, ensuring the datasets are a good test for the generalisability and adaptability of any proposed anomaly detection model.

1.5 Methods of Investigation

Saunders (2011) proposed the research onion model which illustrates the stages of a research project starting with the philosophy behind a research and ending with the data collection methods and strategies. The research onion is a popular model and it is applicable to many research types.

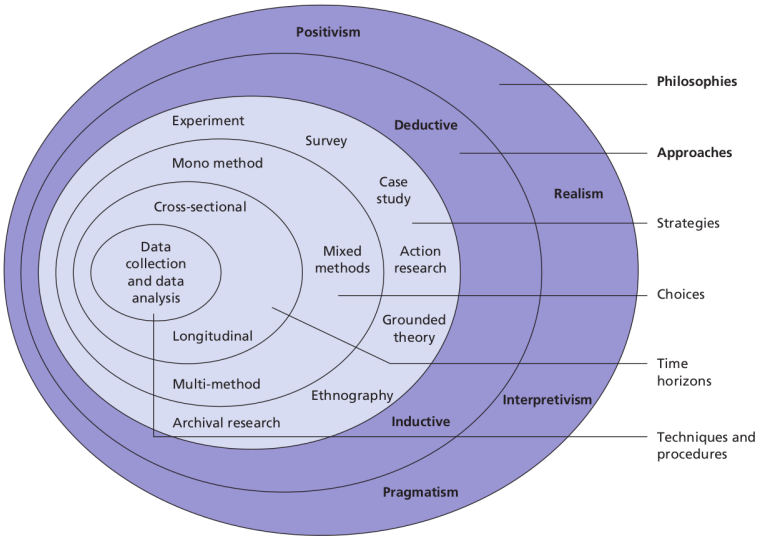


FIGURE 1.2: The research onion model (Saunders, 2011).

As shown in Figure 1.2 The outermost layer of the research onion model deals with the philosophical foundation behind any knowledge pursuit. The positivism view about reality state that reality exists and can be observed and described objectively and independently from the observing subjects (Newman & Benz, 1998). These observations should also be repeatable and applicable. Positivism can also be referred to as empiricism and usually applied to hard science. What is commonly thought of as an opposite to the positivist view is the interpretivist view. The interpretivism view about reality assumes the existence of many realities that are dependant on the interpretation of the observing subjects, and these realities are dependant on time and context (De Villiers, 2005). Interpretivism is also referred to as constructionism and usually applied to social science.

The second layer of the onion model deals with the approaches suitable for every philosophical view of reality. The deductive approach develops a hypothesis based on an existing theory. Based on gathered observations, this hypothesis is either accepted or rejected. Therefore, the deductive approach goes from the general to the specific. The inductive approach, on the other hand, goes from the specific to the general. It starts

by gathering observations. Then, trying to find patterns in these observations to create hypothesis from which a theory will eventually emerge.

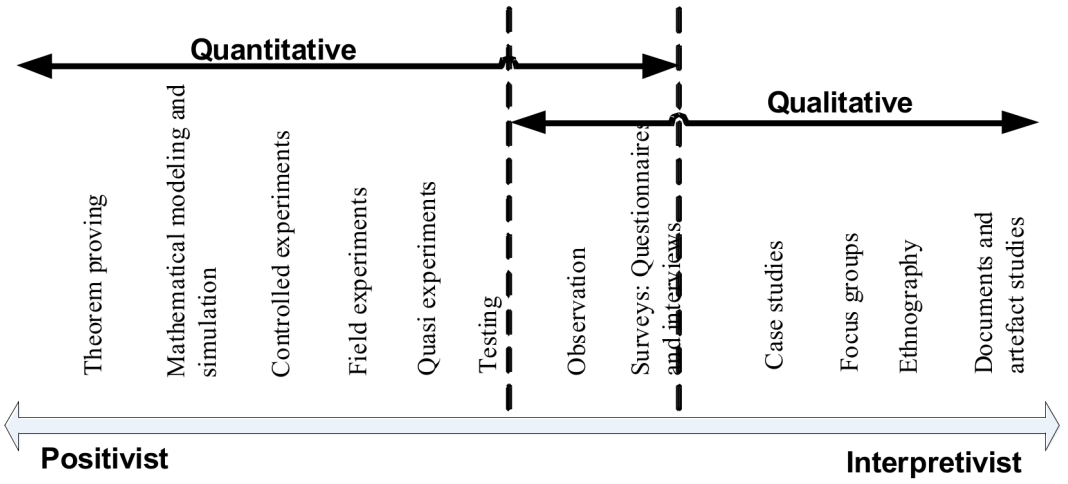


FIGURE 1.3: Methods and strategies of research (De Villiers, 2005).

The choice of data types for a particular research can be categorised into two categories: quantitative and qualitative data types. Quantitative and qualitative research are not mutually exclusive and both can be used in what is referred to as mixed methods. The quantitative research deals with large quantitative data that can be measured and analysed using statistical methods (May, 2011). The quantitative research is usually associated with deductive approaches guided by positivist philosophical views. However, it can be used with other approaches, in social sciences for example, which are usually inductive. The qualitative research is usually associated with interpretivism and inductive approaches and tries to interpret the subject's realities. Figure 1.3 shows the strategies and methods that are suitable for the corresponding data and philosophical views.

This research deals with a big number of quantitative data gathered by simulating the habits and patterns of a smart home inhabitant. This data is statistically analysed and evaluated to measure the performance of an HTM machine learning model at detecting anomalous patterns. The machine learning model is built under the guidance of the HTM theory and the hypothesis that the HTM theory is able to develop a machine learning model capable effectively of detecting anomalies. Therefore, this research follows a positivist view of reality by a deductive approach on quantitative data gathered by simulations.

The main steps taken in this research can be summarised chronologically as follows:

- Conducting a comprehensive literature review of the IoT and smart homes, anomaly detection algorithms and specifically unsupervised biologically-inspired anomaly detection algorithms,
- Studying the HTM theory and its algorithmic implementation and their capabilities in detecting anomalies without supervision,
- Developing a novel smart home simulator, OpenSHS (see Chapter 4), to design a virtual smart home,
- Generating representative synthetic smart home datasets using OpenSHS by volunteering participants,
- Developing a biologically-inspired anomaly detection model based on the HTM theory,
- Developing a novel encoder, HI-SDR (see Chapter 5), that solves the issue with the HTM model and high-dimensional datasets,
- Evaluating and analysing the performance of the novel encoder against the state-of-the-art unsupervised anomaly detection algorithms.

1.6 Structure of the Thesis

This thesis is organised into seven chapters as follows:

- **Chapter 1**: Is an introduction to the thesis that describes the aims and objectives of this work along with the novel contributions to knowledge and methods of investigations used in this research.
- **Chapter 2**: Provides a literature review of the research efforts in the domain of anomaly detection, smart homes, HTM theory and IoT.
- **Chapter 3**: Focuses on the HTM theory and its algorithmic implementation, the CLA.
- **Chapter 4**: Presents OpenSHS, a new smart home simulator, and its implementation details.
- **Chapter 5**: Studies the first region of the HTM models, the encoder. The Chapter also presents the novel encoder "HI-SDR", which is developed to overcome the issue of high-dimensional datasets with HTM systems.

- **Chapter 6**: Evaluates the novel encoder on the forty-two datasets generated by OpenSHS and compares the results with the standard HTM encoders and with state-of-the-art unsupervised anomaly detection algorithms.
- **Chapter 7**: The thesis concludes with this Chapter which summarises the outcomes of the research and the recommendations for future work.

Chapter 2

Literature Review

2.1 Introduction

This Chapter starts with an introduction to the IoT paradigm and its enabling technologies and their issues. The concept of context-awareness is presented and how middleware solutions are used to provide the user with intelligent services. These intelligent services can use Machine Learning models that are capable of detecting anomalies. These models can be used as infrastructure for a middleware solution to provide better context-aware and intelligent services.

Anomaly detection is a well-established field of research. In this Chapter, a literature review of different anomaly detection techniques is presented and critiqued as well as their application domains.

One of the recent advancements in Machine Learning models is the introduction of the HTM theory which is an overarching theory of how the neocortex in the brain works. The theory suggests a more complex model to the neuron than what is typically found in Artificial Neural Networks (ANNs). The Cortical Learning Algorithm (CLA) is the algorithmic implementation of this theory. The theory is briefly presented in this Chapter and studied in details in the next Chapter.

One of the prominent applications of the IoT vision is the smart home. A definition of what constitutes a smart home, is presented in this Chapter as well as its applications. The role of anomaly detection in the smart home and what it can provide is also presented.

The Chapter is concluded by an initial study of the CLA on a small dataset generated by a prototype smart home simulation tool (which later became OpenSHS, see Chapter 4) and the results are compared to the DBSCAN clustering algorithm.

2.2 The Internet of Things

Kevin Ashton first coined the term "Internet of Things" in 1999 (Ashton, 2009) to express the idea that all of the information on the Internet today is captured by humans and by having smart objects[1] assessing the humans in capturing or even taking the role of capturing the information. The research of Auto-ID Labs (2003) in the Radio-frequency identification (RFID) field gave the IoT popularity and attention. Since the IoT research field is evolving, there are different definitions of the term. Tan & Wang (2010) defined it as: "Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts". Also, according to Vermesan *et al.* (2011): "The Internet of Things could allow people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service". In this research proposal, the author believes that the latter definition captures the overall picture of the IoT future vision.

2.2.1 IoT Issues

The realisation of the IoT ultimate vision is faced with many obstacles. As with any new technology, standardisation is one important obstacle to overcome. For instance, in the RFID field, the Electronic Product Code (EPCglobal) standard by Global Standards One (2003) promotes a global adaptation of the Electronic Product Code (EPC). The IEEE 802.15.4 (Molisch *et al.*, 2004) is the physical and Media Access Control (MAC) layer standard for low-rate wireless personal area networks which is the basis for the Internet Engineering Taskforce (IETF) IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) standard (Kushalnagar *et al.*, 2007). 6LoWPAN allows IPv6 packets to be transferred over networks following the IEEE 802.15.4 standard. The same IEEE 802.15.4 standard is built upon by proprietary technologies like ZigBee (ZigBee, 2006). Such proprietary solutions will hinder the realisation of the IoT vision and will contribute greatly to the fragmentation and lack of standardisation. The success of the current Internet could be attributed to its transparency and openness which is important to a worldwide network used by billions of users. The necessity of open standards is a crucial aspect of a network that will span everything around us.

In addition to the standardisation issue, a survey by Atzori *et al.* (2010) identified other technical issues such as the addressing of the things in the IoT. Since the number of the connected things will be huge, IPv6 is an appealing choice, but due to the different incompatible technologies implemented, more effort is needed in this regard. The

[1] Smart objects, things and devices are used interchangeably.

6LoWPAN standard is an example of such efforts. Furthermore, there is a need for a naming service analogous to the Domain Name System (DNS) of the current Internet. Concepts like the EPCglobal's Object Naming Service (ONS) (Brock, 2001) could be valuable. Also, the scalability and reliability of the network is another important issue to be tackled taking into consideration the demands of things in the IoT paradigm.

The research activities in the IoT field regarding the Internet network stack itself could be categorised into two categories. One category promotes the use of the existing Internet infrastructure and standards without any modification and that stance is led by the Web of Things (WoT) research (Guinard, 2011; Trifa, 2011). The other category proposes the change or modification of the Internet infrastructure. An example of the latter is the Constrained Application Protocol (CoAP)(Bormann *et al.*, 2012) which is a lighter alternative to the Hypertext Transfer Protocol (HTTP).

Security in the IoT context is another major open issue. With the abundance of wirelessly connected devices, issues like the man-in-the-middle attack could be a real threat (Li *et al.*, 2012). Furthermore, the sheer number of connected devices could make Distributed Denial of Service (DDoS) attacks trivial to achieve. Bandyopadhyay & Sen (2011) identified several open security issues such as securing the IoT architecture and the proactive identification of malicious software and attacks. Considering the huge number of connected things in the IoT paradigm, privacy and ownership management is another important open issue. In the same paper, the researchers identified several challenges such as data privacy, location privacy and tools for identity management of users and objects.

The IoT paradigm requires the ability for anything to connect and communicate using any available means. This requirement was identified by many researchers and typically referred to as the problem of heterogeneity of the things and lack of interoperability (Haller *et al.*, 2009).

2.2.2 Enabling Technologies

The IoT vision could be a reality due to advancement in the RFID and Wireless Sensor Networks (WSN) fields. For instance, a group of researchers (Welbourne *et al.*, 2009) managed to build a miniature IoT network using RFID technology and conducted a study for the users' reactions. Fosstrack (Floerkemeier *et al.*, 2007) is an open source RFID platform that implements the GS1 EPC standards. Also, WSN could play a major role in the IoT. Molla & Ahamed (2006) presented ten key obstacles that must be considered when building a middleware for sensor networks and also the researchers conducted a survey of existing middlewares. Mohamed & Al-Jaroodi (2011) surveyed

service-oriented middlewares for WSN. Gubbi *et al.* (2013) showed how RFID and WSN are important enabling technologies and predicted they will play an important role for the future IoT.

2.2.3 Context-awareness

Due to the enormous number of smart devices envisioned in the IoT paradigm, managing and controlling each smart device individually will be cumbersome for the user. The smart device's interaction with the user should not be intrusive and should be intelligent and able to adapt to the user's habits. Moreover, the interaction should take into account the current situation of the user. Awareness of the present environment of the user is called context-awareness which is a term first coined by Schilit & Theimer (1994) in 1994. Dey *et al.* (2001) surveyed and critiqued the different definitions of the term. Perera *et al.* (2014) conducted a comprehensive survey of context awareness from the IoT perspective and 50 projects from the last decade were compared and critically evaluated to conclude lessons for the future research in this field. Moreover, the study conducted a comparison of the AI techniques used in different middlewares to model and represent context. According to Dey *et al.* (2001), the definition of Context is: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Also from the same paper, the definition of Context-awareness is: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." (Preuveneers & Berbers, 2008) identified context-awareness as a key part to the realisation of the IoT ultimate vision.

2.2.4 Middlewares

Issarny *et al.* (2007) defines the middleware as software that acts as a mediator between the applications and the distributed operating system providing services to well-known issues such as heterogeneity, security and interoperability. The middleware could be looked at as an operating system which allows software applications to interact with the smart devices via an Application Programming Interface (API). The middleware layer at hand tries to solve some of the issues facing the adoption of the IoT paradigm in domestic smart home settings. Mainly, the management of the vast number of the connected devices and detecting the user's usage patterns.

The literature is full of projects and research activities each with its own focus and scope. One interesting work was presented by Kawsar (2009); Kawsar & Nakajima (2009) where a document based and user-centric framework for smart objects was proposed. The document based approach was influenced by the design of the World Wide Web, which enabled the exchange of information in a platform agnostic way. The framework utilises this approach to overcome the heterogeneity problem of smart objects. Smart objects present their services in XML documents to a governing body called FedNet. FedNet manages the requests coming from the applications and connect them with the available services of smart objects. Furthermore, the framework presented user-centric tools built on top of the system to aid in the management of the users' devices. The framework also presented abstractions and design methodologies helpful for the application developers and manufacturers. The framework, however, could be improved in many ways to meet the needs of the IoT vision.

Soma Bandyopadhyay *et al.* (2011) conducted a survey and a comparison of existing IoT middlewares. Some of the comparison factors were the device management, interoperation, context-awareness, security and privacy. The survey showed a lack of interest in the context-awareness aspect and more work could be done to improve the state of the current IoT middlewares.

The open source Robot Operating System (ROS) (Quigley *et al.*, 2009) was successfully used in a study by Kranz *et al.* (2010b) to build a smart office environment where everything in the office can send updates via Twitter. Contiki (Dunkels *et al.*, 2004) is also a popular tiny, lightweight and event-driven operating system suitable for devices with scarce resources.

Previous work was done by Roalter *et al.* (2010) to use ROS as an IoT middleware and the simulations tools of ROS were used to create an intelligent environment called the Cognitive Office.

There are many methods to model context. Each method has its strengths and weaknesses. Perera *et al.* (2014) surveyed some of the ways to model context such as:

- Key-Value pairs
- Markup and tags
- Graphical-based
- Object-based
- Logic-based

- Ontology-based

Lim & Dey (2010) conducted a survey of 109 context-aware applications and the AI techniques used in these applications. Figure 2.1 shows a pie-chart of traditional AI techniques used in these applications. The majority of these context-aware applications use simple conditional statements to construct rules.

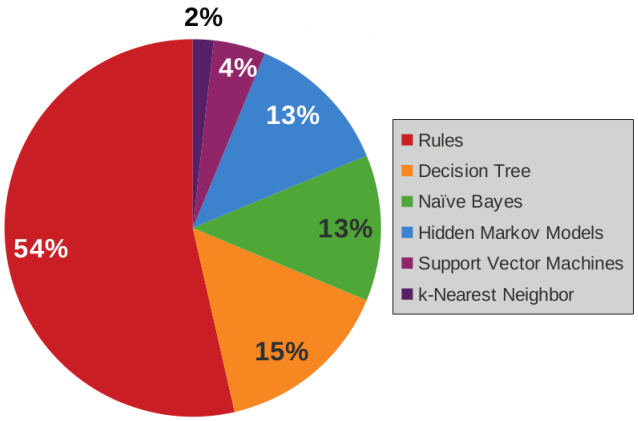


FIGURE 2.1: AI techniques used in 109 context-aware applications (Lim & Dey, 2010).

In a study conducted on 192 participants, the author explored some of the participants habits with regard to their smart devices. 68% of the participants reported that they have a smart device other than a smart phone. Most of these smart devices are tablets (81.4%). When asked if the participant would like to have a software that manages all of their smart devices, 70.7% said yes. This indicates that having a middleware to manage the smart devices is a popular choice (Alshammari *et al.*, 2015).

2.3 Intelligence

The AI field is one of the oldest fields in computer science history. AI was pioneered by prominent figures such as Alan Turing, Marvin Minsky and John McCarthy. AI as an academic discipline started in the 1950's and John McCarthy defined the study of AI as 'the science and engineering of making intelligent machines' (McCarthy, 1998). The English mathematician and computer scientist Alan Turing laid some of the theoretical foundations of AI by introducing the Turing test in his seminal paper 'Computing Machinery and Intelligence' (Turing, 1950). The Turing test is devised to assess the existence of intelligence in a machine. This consists of three parties, a human interrogator, another human in an isolated room and a machine in another isolated room. The interrogator communicates with the two parties through written text in a natural language (or any neutral communication means such as a computer screen). If the interrogator is not able

to successfully identify the responses of the machine from the human, then the machine is then said to have intelligence.

Jeff Hawkins in his book 'On Intelligence' (Hawkins & Blakeslee, 2004) criticises the traditional view of intelligence in the AI field. To Hawkins, intelligence was not formally defined, and the AI field only looks at the brain as a *computing* machine. A paper published in 1943 by McCulloch & Pitts (1943) described how it is possible for the neurons to operate as logical gates. This paper helped in solidifying the view that the brain is just a computational machine similar to a Turing machine. At the beginning, the AI research seemed the correct approach to achieve true intelligence, and many success stories were happening such as IBM's Deep Blue (Campbell *et al.*, 2002) beating Gary Kasparov in a game of chess. It was claimed that what kept AI from reaching its full potential was just a limitation of hardware and speed according to traditional AI research. But nevertheless, the AI winter came, and the field seemingly reached a dead end. The success stories of these intelligent machines were very specific to a particular domain and application. AI did not provide a flexible and human-like intelligence and over time the need for alternative approaches to achieve true intelligence became pressing.

From a philosophical point of view, this idea of viewing what the brain does as simply computations and the intelligence of the brain can only be measured and assessed by the behaviour it produces, this find its roots in the philosophical theory of behaviourism. For the behaviourist, all that can be known about a state of mind is through its behaviour. This philosophical idea gave birth to other schools of thought in the philosophy of mind fields such as functionalism and computationalism.

John Searle famously criticised this view and came up with a thought experiment as an argument against the behaviourist view (Searle, 1980). In Searle's Chinese Room argument, Searle proposes the idea of him being in a locked room, and he receives a letter written in Chinese which he does not understand nor speak. Along with the letter he also receives another letter written in English which contains instructions on how to manipulate these Chinese symbols in certain ways. According to Searle, the Chinese symbols are just abstract symbols that do not mean anything to him. So he follows the instructions in the English letter which instructs him to manipulate the Chinese symbols by writing a symbol in one place and erasing another and so on. He follows the instructions blindly until the last instruction that tells him to submit the Chinese letter. A Chinese outside observer takes the letter and reads it. The Chinese letter was a story and there were some questions which Searle had to answer. To the Chinese outside observer, the answers are correct and indicate to him that Searle did *understand* the story because he correctly responded to the questions. Searle on the other hand, did not understand what the story is about. He had blindly followed the instruction letter.

Searle explains that the English letter is similar to a software program; Searle was the CPU executing blindly the instructions set without real understanding of what the Chinese story really means, this intelligent behaviour is called *weak AI*. If the story letter was in English and Searle *understood* the meanings of the English words and was able to answer the questions, then he calls this behaviour *strong AI*.

Searle did not offer an explanation of what it means to be an intelligent machine. His objection was that no Turing machine will be able to produce real and human-like intelligence. Russell Stuart and Peter Norvig in their book (Russell & Norvig, 2009) stated that most of AI research accepts the weak AI hypothesis and does not care about whether what their program doing is really intelligent or just a mere simulation of intelligence. The important thing is that the program is able to do what it is suppose to do.

An interesting research initiated by Jeff Hawkins (Hawkins & Blakeslee, 2004) takes a drastically different approach to intelligence than traditional AI techniques. Hawkins argues that traditional AI was built on a false premise. The premise that the brain does *compute* and that intelligence is fundamentally just *computations*. According to Hawkins, traditional AI research was built on the work of Alan Turing who never formally defined what intelligence really is. For Hawkins, Alan Turing proposed an existence test for intelligence, the famous Turing test, but he did not worry about the nature of intelligence and what intelligence means. To Hawkins, this approach is what hindered traditional AI from reaching its full potential. The argument that the *brain is much faster at computing* which traditional AI proponents believe is simply not true. With the advancements in Neuroscience, the brain seems slower than the computers that we have today. Hawkins praised some of the work done in neural network's research but he argues that even some of the neural networks research suffers from an over-simplified simulation of what really happens in the brain. Also, the neural network's research still views intelligence fundamentally as *computations*.

2.4 Anomaly Detection

Anomalies are rare data points that are different from the majority of the data. (Hawkins, 1980) Defined anomalies formally as:

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

The term anomaly is not the only term used in the literature. *Outliers, abnormalities and deviants* are other terms used in the literature to mean the same thing. Anomaly

detection techniques are used when the anomalies are of particular interest to the user. For example in a bank account, the transactions pattern is usually predictable and when an anomalous transaction happens, this is probably an indication of an unauthorised use of the bank account. There are many applications for anomaly detection, such as: credit card fraud, intrusion detection, medical diagnosis, earth science, law enforcement and sensor networks attacks (Aggarwal, 2013).

2.4.1 Challenges

The anomaly detection problem can be considered abstractly as identifying patterns or data points that do not lie in a normal region. Therefore, the problem can be approached by recognising a normal region and anything outside this normal region is flagged as an anomaly. However, the problem in practice is very challenging (Chandola *et al.*, 2009). Here are several issues that explores the difficulty associated with anomaly detection:

- Defining what is *normal* is hard and usually there is no clear distinction between what can be considered normal and abnormal,
- The *normal* behaviour is ever changing and evolving and different from one domain to another,
- Anomalies have various forms and types specific to the domain they reside in,
- Availability of representative datasets with or without labels,
- Distinguishing true anomalies from noisy data.

Therefore, the task of anomaly detection is not trivial and varies from domain to domain. Chandola *et al.* (2009) claim that most of the existing anomaly detection techniques are formalised to solve a particular domain problem and are not generally applicable to other domains.

The nature of the anomalies is an important aspect in anomaly detection. Anomalies can be categorised into three types:

- **Point anomalies:** Where anomalous data points are regarded so different from the rest of the data. In Figure 2.2, regions N_1 and N_2 are considered normal because most of the data points are in these two regions. On the other hand, O_1, O_2 and O_3 are far from the normal regions and considered anomalies.

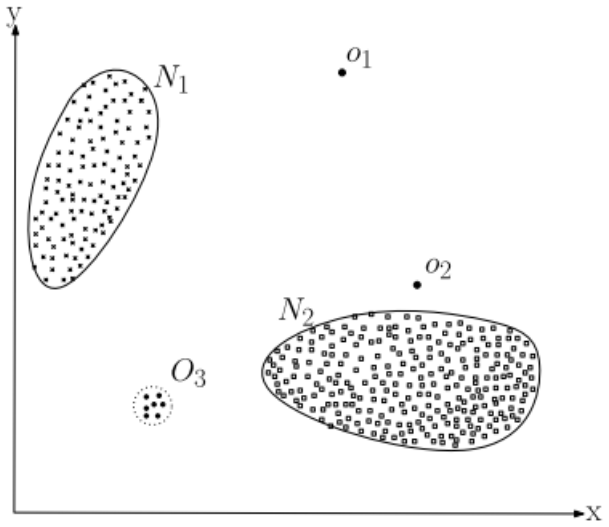


FIGURE 2.2: Point anomalies in a two dimensional space (Chandola *et al.*, 2009).

- **Contextual anomalies:** Where the context of the data points is anomalous and not the data point itself. As shown in Figure 2.3, the data point t_1 is identical to the data point t_2 but the latter point is considered anomalous because it appears in an anomalous context.

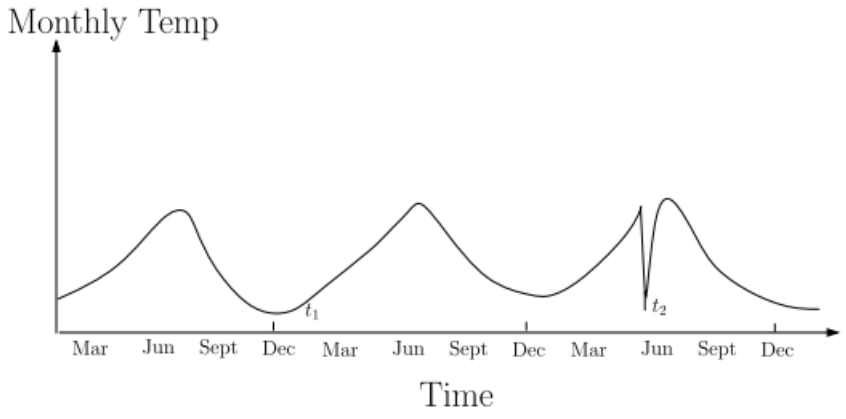


FIGURE 2.3: Contextual anomalies in a monthly temperature data (Chandola *et al.*, 2009).

- **Collective anomalies:** Where a collection of data points is considered anomalous not because of the data point themselves but because of the *collection* of these data points together. In Figure 2.4, The data points in the electrocardiogram are considered anomalies because of their appearance as a collection in this data and not because of the data points themselves.

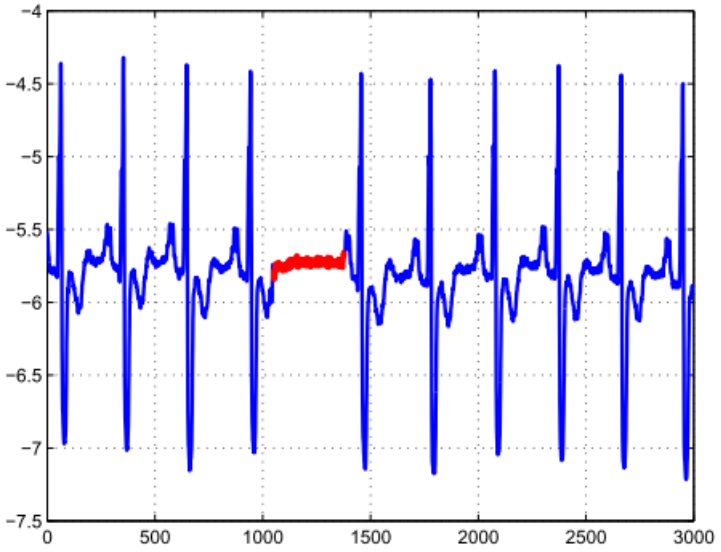
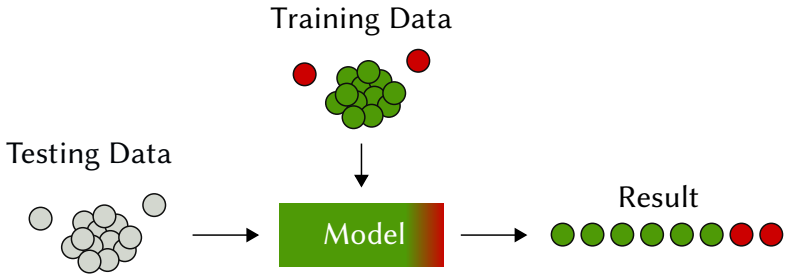


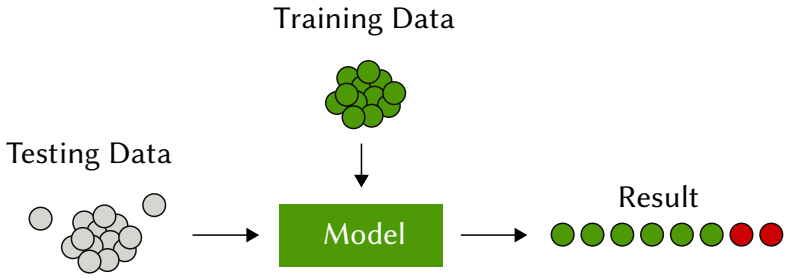
FIGURE 2.4: Collection anomaly in a human electrocadriogram (Chandola *et al.*, 2009).

Another important aspect in anomaly detection is the availability of data labels. It is possible to categorise anomaly detection techniques with regards to the availability of the data labels into three categories:

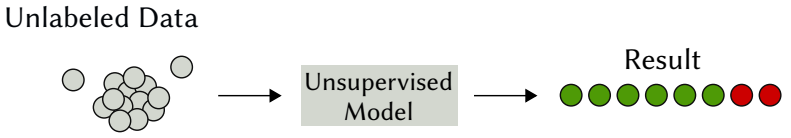
- **Supervised:** The assumption here is that there is a training data set with labels identifying normal and abnormal data points.
- **Semi-supervised:** The assumption here is that the available training data is all normal and the deviation from these normal data points is considered an anomaly.
- **Unsupervised:** The assumption here is that the data is not labelled and no training data is needed. The techniques in this category assume that the majority of the data points are normal and thus, these techniques group or cluster the data points into clusters and any isolated points are considered anomalies.



(A) Supervised anomaly detection.



(B) Semi-supervised anomaly detection.



(C) Unsupervised anomaly detection.

FIGURE 2.5: Anomaly detection and the approaches with availability of training and testing data (Goldstein & Uchida, 2016).

2.4.2 Anomaly Detection Techniques

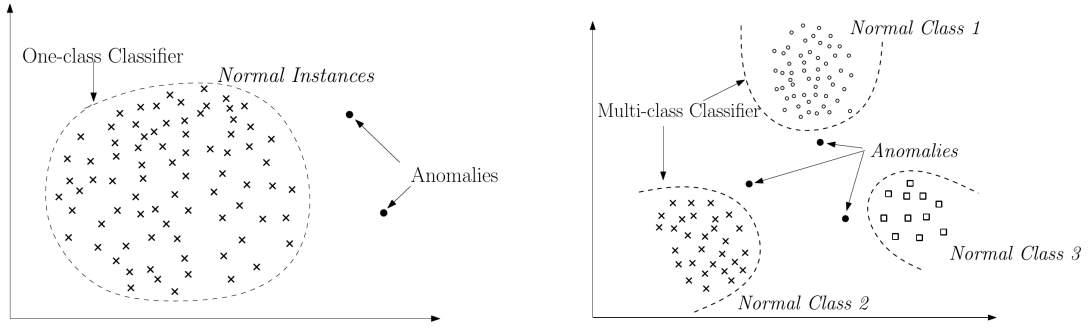
In this Section, several approaches and techniques to detect anomalies will be presented. The fundamental technique that each approach is based on will be the categorisation factor used.

2.4.2.1 Classification

The techniques here need labelled data points for the models to learn from. The main idea is to train a classifier on the normal data points and then evaluate the accuracy of the model on never been seen data points, called the testing data points. From the perspective of how many classes can be learned, these techniques can be further divided into two categories: *one-class* and *multi-class* anomaly detection techniques. The one-class assumes that the training data points are all normal. Therefore, the model learns the characteristics of these data points and classify them as normal. Any data point that does not fall into this normal class, will be classified as an anomaly by the model. As

shown in Figure 2.6a, the one-class model will group all normal data points as one big class and any points residing outside this class are flagged as anomalies. To learn the normal region, several algorithms can be used. One popular algorithm is the Support Vector Machine (SVM) (Schölkopf *et al.*, 2001; Heller *et al.*, 2003; Manevitz & Yousef, 2001). Roth (2006) used Kernel Fisher model to learn the normal class.

The multi-class category is similar to one-class except that instead of learning one normal region, multiple regions can be learned, as shown in Figure 2.6b. Barbara *et al.* (2001); De Stefano *et al.* (2000) used multi-class classification techniques to detect anomalies.



(A) One-class classification. (B) Multi-class classification.

FIGURE 2.6: Classification based anomaly detection techniques (Chandola *et al.*, 2009).

The classification techniques can be organised based on the algorithm used by the model into several categories, Support Vector Machines, Artificial Neural Networks, Bayesian networks and rules.

The SVM (Vapnik, 2013) can learn the normal region from the training data samples and with the *kernel trick* (Boser *et al.*, 1992) the model can lean non-linear regions in hyperplanes. Extending the algorithm with kernels like the Radial Basis Function (RBF) kernel allows the model to learn complex regions. Many extensions to the support vector machine have been proposed to be applied in the context of anomaly detection. King *et al.* (2002) proposed a technique to detect novelty in power plants. Davy & Godsill (2002) extended the SVM algorithm to identify anomalies in audio signals. The SVM algorithm is a popular technique to detect computer host intrusions as done by Heller *et al.* (2003); Davy & Godsill (2002); Lazarevic *et al.* (2003). Ma & Perkins (2003b,a) applied SVMs in temporal sequences of data. Song *et al.* (2002) proposed a variant of SVM called robust SVM (RSVM) which is more robust to anomalies during the training phase. Hu *et al.* (2003) applied RSVM to compute host intrusion detection.

Artificial neural networks can be applied to one-class and multi-class classification problems. The technique trains a neural network model on a portion of the dataset. Then,

the testing data points are fed to the network and when the network rejects the data point, it is flagged as an anomalous data point (De Stefano *et al.*, 2000; Taylor & Addison, 2000). Many different types of neural networks have been used in the literature. Augusteijn & Folkert (2002) used Multi-Layered Perceptrons (MLP) to detect novelties. Thompson *et al.* (2002); Diaz & Hollmén (2002) used variations of auto-associative networks to detect anomalies. Crook & Hayes (2001); Crook *et al.* (2002); Murray (2001) used Hopfield neural networks. Williams *et al.* (2002); Hawkins *et al.* (2002) applied Replicator Neural Networks (RNNs) for one-class anomaly detection.

Bayesian networks based techniques can be applied for multi-class classifications. These techniques can be used with univariate and multivariate datasets. The main idea is to use a Naive Bayesian network and train it on a training set to estimate the prior probabilities. Then, a testing set of data points will be fed to the network and for each data point, the data point with the biggest posterior score will be the class or the label. Sebyala *et al.* (2002); Barbara *et al.* (2001); Valdes & Skinner (2000) used Bayesian networks to detect intrusions in computer networks. Diehl & Hampshire (2002) applied Bayesian networks to identify anomalies in surveillance footage. Baker *et al.* (1999) applied Bayesian networks in the context of detecting anomalies in textual datasets and (Wong *et al.*, 2003) for detecting diseases outbreaks.

Anomaly detection techniques that are based on rules learn from normal data points and any data point that does not conform to any learned rule, is considered an anomaly. The main idea in this technique is to train a model on a training set so the model can derive rules. Every rule is assigned a *confidence* score, a ratio of the correctly classified data points by this rule. During the testing phase, the model will search for the best matching rule for each testing data point. The anomaly score for each testing data point will be the inverse of the confidence score of the best matching rule. Decision Trees and association rule learning are popular rule based models in the literature. Salvador *et al.* (2004); Fan *et al.* (2004) proposed extensions to fit rule learning to the context of anomaly detection. Tandon & Chan (2007); Chan *et al.* (2003); Mahoney & Chan (2003); Otey *et al.* (2003); Mahoney & Chan (2002) used association rules to detect computer networks intrusions. He *et al.* (2004) proposed an anomaly detection technique for categorical datasets. Yairi *et al.* (2001) used a rule based technique to detect fraud.

The previously presented classification techniques have several advantages. One of the advantages of these techniques is the abundance of classification models that will give the research the ability to choose an appropriate model to the problem at hand. Another advantage is that once a model is trained, it is usually computationally fast to test and query a data point to classify it as an anomaly or not.

On the other hand, these approaches have several disadvantages. One of the disadvantages is the availability of correctly annotated training datasets which are difficult to obtain in some domains. Another disadvantage is that the output of these techniques is a label and not an anomaly score. This can be restrictive for further elaboration on the dataset by other models that rely on scalar values and not binary values.

2.4.2.2 Nearest Neighbours

The anomaly detection techniques in this category rely on computing a distance measure between two data points and based on this distance, the data points are organised in neighbourhoods to understand the structure of the dataset. The distance measures used depend on the type of the variables or the feature space. Usually for numerical data points, the Euclidean distance is used (Tan *et al.*, 2006). For categorical features, Jaccard distance can be used and other methods (Chandola *et al.*, 2008). The algorithms in this category can be further divided into two sub-categories: algorithms based on the K^{th} nearest neighbour and algorithms based on the density of the data points.

The K^{th} nearest neighbour distance can be used as an anomaly score for a collection of data points. Guttormsson *et al.* (1999) used $k = 1$ nearest neighbours to detect anomalies in the operation of turbine motors. A threshold can be set by a field expert to separate anomalous data points from normal ones. This basic idea saw several improvements and extensions in the literature. Zhang & Wang (2006a) defined the anomaly score as the sum of the data point's distances to its k nearest neighbours. Bolton *et al.* (2001) used similar anomaly score definition to detect fraud in credit card transactions. Another technique used in the literature is to simply count the data points that are in a neighbourhood of a specific distance (Knorr & Ng, 1997; Knorr *et al.*, 2000). Other techniques and extensions were proposed for dealing with categorical data types such as Kou *et al.* (2006); Otey *et al.* (2006); Palshikar (2005); Wei *et al.* (2003).

The density based techniques measure the density of data points neighbourhoods. Any data point that resides in a low density neighbourhood is flagged as an anomalous data point. The techniques here rely on having close to uniform densities for the data points. Figure 2.7 shows sample data points with different densities to illustrate this issue. To a human observer, it is obvious that point $p1$ and $p2$ are anomalies. The simple density based techniques will fail to flag $p1$ as an anomaly because the distance between $p1$ and the neighbourhood/cluster $C2$ is shorter than the distances of the points in $C1$. To mitigate this issue, Breunig *et al.* (2000) proposed the popular Local Outlier Factor (LOF) density based algorithm that takes into account the ratio of the average density in each neighbourhood or cluster. To calculate the density of a neighbourhood, a radius

of a small hypersphere is defined for a neighbourhood with a data point at its centre. Then, the hypersphere volume is divided by the number of data points in the neighbourhood and the resulting number is the density score for this neighbourhood. The anomalous data points can be easily defined as the data points that reside outside of the neighbourhood. Therefore, LOF can detect that $p1$ and $p2$ are anomalies in Figure 2.7.

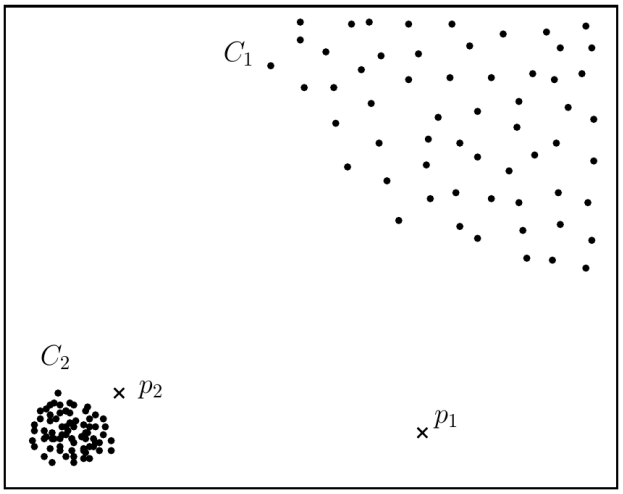


FIGURE 2.7: Data points with varying densities (Chandola *et al.*, 2009).

Many extensions in the literature to the LOF algorithm were proposed that improve the calculation time and reduce the complexity of the algorithm. Hautamaki *et al.* (2004) proposed Outlier Detection using In-degree Number (ODIN) which simplifies the density calculation. Some extensions change the way the density calculation is performed. One popular extension is the Connectivity-based Outlier Factor (COF) that was proposed by Tang *et al.* (2002). COF computes the members of a neighbourhood incrementally and step by step. Starting with the closest data point and gradually adding new data points until the neighbourhood data points reach the k^{th} size. Figure 2.8 shows the difference of how this calculation is performed. The anomaly score calculation is done the same way it is done in LOF.

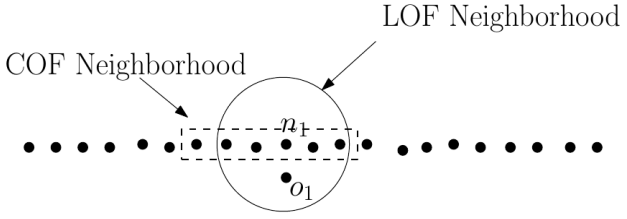


FIGURE 2.8: LOF and COF density measurement (Chandola *et al.*, 2009).

The nearest neighbour techniques have many advantages. One of the attractive points to use these techniques is when the anomaly detection problem requires the operation in an unsupervised fashion. From what was presented, these techniques try to arrange and assemble the data points in groups or clusters based on a certain distance measure. Therefore, they do not require the availability of target labels for every data point. Another advantage of using nearest neighbours techniques is that they can work with any type of data assuming there is an appropriate distance measure defined for each data type, they also can work in a semi-supervised fashion.

The main disadvantages of these techniques is that they rely on the normal data to have a cluster structure. Moreover, these techniques can be computationally expensive depending on how the distance is calculated. Some of the techniques have $O(N^2)$ complexity in the number of data points since the distance for each data point is calculated against all data points.

2.4.2.3 Clustering

Clustering based anomaly techniques have many similarities to the nearest neighbours techniques. The nearest neighbours techniques perform the calculation between a data point and its local nearest neighbour. While the clustering based techniques perform the calculation between each data point and the group or the cluster that it belongs to; based on a similarity measure. Clustering techniques work in an unsupervised fashion and can also work in a semi-supervised fashion. A family of the clustering techniques assume that the normal data has a cluster and any data point outside of this cluster is flagged as an anomalous data point. Usually the main objective of the clustering algorithms is to find structures in datasets. Thus, they are used to perform exploratory analysis of datasets and are used in recommender systems. Some of these clustering techniques allow data points to reside outside the cluster they build. The algorithms that allow for this condition can be used to perform anomaly detection. Examples of these techniques are DBSCAN which was proposed by Ester *et al.* (1996), ROCK (Guha *et al.*, 2000) and SNN (Ertöz *et al.*, 2003). Due to the nature of how these algorithms work, the output of these models is binary.

Another family of these techniques perform under the assumption that normal data points are organised around the cluster centre or centroid. The data points that are not close to the centre are identified as anomaly data points. The general procedure of these techniques starts by using a clustering algorithm to group the data points. Then for every data point, the distance from the data point to the cluster centroid is defined as the anomaly score. Several algorithms were successfully used to achieve this goal

such as Self-Organising Maps (SOMs) (Kohonen *et al.*, 2001), Expectation Maximisation (EM) and K-means algorithms (Smith *et al.*, 2002).

The third family of these clustering techniques assumes that normal data points reside in dense clusters and anomalous data points are grouped into low density clusters. There are many algorithms in the literature that follow this assumption. Examples of these algorithms are Cluster-Based Local Outlier Factor (CBLOF) which is proposed by He *et al.* (2003), Pires & Santos-Pereira (2005) and Jiang *et al.* (2001). Similar to the other families, many extensions to these algorithms were proposed in the literature to improve the performance of these algorithms (Sun & Chawla, 2004; Eskin *et al.*, 2002; Chaudhary *et al.*, 2002; Portnoy *et al.*, 2001).

The advantages of the clustering based techniques are similar to the nearest neighbours techniques as both can operate in an unsupervised and semi-supervised fashion. They also can work with many data types. Much like the nearest neighbours techniques, the performance of these techniques relies on the performance of the clustering algorithm used. Moreover, computational complexity can be high. Some of the clustering techniques are not mainly geared towards anomaly detection but they are extended to perform anomaly detection which can make the performance of these approaches less than optimal.

2.4.2.4 Statistical

The statistical approaches to detect anomalies are a well established and old research field. This quote from Anscombe (1960) defines what anomalies means in the statistical approaches:

> "An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed "

Statistical techniques try to fit a statistical model on the normal data points distribution. Then, statistical tests can be used to identify whether a data point is normal or anomalous. The statistical techniques for anomaly detection problems can be divided into two categories, parametric and non-parametric techniques.

Parametric techniques assume the existence of a distribution and its parameters can be learned from the data points. The distribution parameter is referred to as Θ, which is estimated from the training data points, and the probability density function (pdf)

is $f(x, \Theta)$ for any given data point x. The anomaly score is defined as the inverse of $f(x, \Theta)$. This category can be classified based on the distribution model used.

One of the popular distributions is the Gaussian model. The parameter for this model can be calculated by the Maximum Likelihood Estimates (MLE) and the anomaly score is defined as the distance of the data point from the distribution mean. When the model is defined, simple thresholds can be applied to filter out the normal data from the anomalous data. One of the oldest works that use this simple approach is Shewhart (1931). Examples of more elaborate works are Beckman & Cook (1983); Barnet (1976); Barnett & Lewis (1964). If the data is normally distributed, Grubb's test can be used to detect anomalies (Grubbs, 1969; Stefansky, 1972). Grubb's test can be used with univariate datasets. An extension to this statistical test was proposed by Laurikkala *et al.* (2000) to make it work with multivariate datasets. The student t-test has been used to detect anomalies by Surace *et al.* (1998). Chi-squared (χ^2) test was used by Ye & Chen (2001) to detect intrusions. A mixture of several Gaussian models or any parametric model can also be used to detect anomalies (Agarwal, 2007; Yamanishi *et al.*, 2000).

Another category of parametric techniques uses a regression model to detect anomalies. These techniques usually are used in time-series analysis (Abraham & Chuang, 1989; Fox, 1972). These techniques fit a regression model based on the training data points and then the anomaly score for a data point is calculated as how far this testing data point is from the regression model. Rousseeuw & Leroy (2005) proposed the popular robust regression model which can deal with anomalies present in the training dataset. Another robust regression models that has been proposed on Auto-Regressive Integrated Moving Average models (ARIMA) (Da *et al.*, 2005; Bianco *et al.*, 2001; Lauer, 2001).

Non-parametric techniques do not assume the existence of a distribution of the data points. Rather, distribution of the data is derived from the data points. These techniques can be further divided into histogram based and kernel function based techniques. For the histogram approaches, a histogram is generated from the training data points. Then for the testing data points, a test is performed to determine if the data point lies in one of the histogram bins or not. If it does, then it is flagged as a normal data point. Otherwise, it is flagged as an anomalous data point. Examples of this approach are Dasgupta & Nino (2000); Helman & Bhangoo (1997); Anderson *et al.* (1995) and for multivariate datasets Yamanishi *et al.* (2000); Manson (2002); Kruegel & Vigna (2003).

Kernel based techniques use kernel functions to estimate the density of the dataset. These techniques are similar to the parametric techniques shown earlier with the exception of how the density function is computed. Examples of these techniques are Yeung & Chow (2002); Tarassenko *et al.* (1995); Bishop (1994).

Statistical approaches in general have several advantages. When the dataset distribution is known, many algorithms and options are available to perform anomaly detection. The output of the anomaly detection models is a scalar value which allows for more sophisticated approaches to be carried out based on this score. Most of these techniques can operate in a supervised and semi-supervised fashion. However, some techniques that can deal with anomalies in the training datasets can also work in an unsupervised fashion.

The main disadvantage of statistical approaches is the reliance on the existence of a distribution of the datasets which is often not the case in real-world data. Even if there is a known distribution of the data points, finding the appropriate statistical test can be a challenge. Multivariate datasets can be a difficult task especially for histogram based techniques.

2.4.2.5 Spectral

Spectral techniques or subspace anomaly detection techniques try to capture a meaningful representation of the data points by reducing the dimensions of the datasets to lower dimensions that could reveal structures not visible in the original form of the dataset. The reduction step is also referred to as embedding or projecting the data point to lower dimensions. These techniques work in an unsupervised fashion and can be used in conjunction with other models. They can also be used to perform pre-processing for the data points before feeding the data to the model.

The prominent algorithm in this category is the Principal Component Analysis (PCA) (Jolliffe, 2002). Agovic *et al.* (2008) used PCA to project the data points to lower dimensions where it is easy to identify the anomalies. Sun *et al.* (2007) developed Compact Matrix Decomposition (CMD) to detect anomalies in matrices. Idé & Kashima (2004) used spectral techniques to identify anomalies present in time series datasets. Shyu *et al.* (2003) proposed an extension to PCA called robust PCA that is able to calculate the principal components using the covariance matrix of the training data points.

One of the main advantages of using spectral techniques is their ability to handle high dimensional datasets. Reducing the feature space to lower dimensions makes it easier for the model to learn the characteristics of the data. Thus, spectral techniques can be used with other models that cannot handle high dimensionality. The ability for the spectral techniques to operate in unsupervised fashion is also another advantage point. On the other hand, these techniques can only perform well if the data is separable when projected to lower dimensions. Another disadvantage is that they can be computationally costly especially when dealing with big datasets.

2.4.3 Unsupervised Anomaly Detection Algorithms

In this Section, unsupervised anomaly detection algorithms will be presented because of their relevancy to this thesis. The main objective of unsupervised anomaly detection techniques is to arrange the data points into groups or clusters in a way that allows the algorithm to identify data points that are isolated and deviating from the normal clusters. This operation is *unsupervised* because there are no target labels for each data point fed to the model to learn from and draw associations. These models are completely driven by the data points they receive.

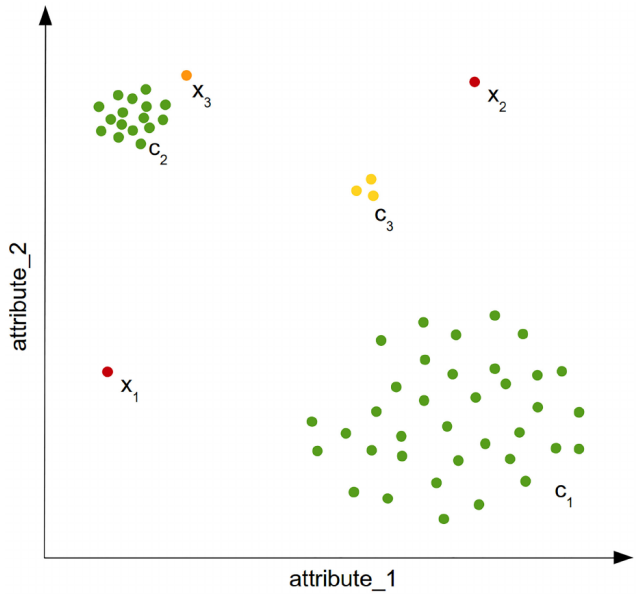


FIGURE 2.9: The global anomalies x_1, x_2 and the local anomaly x_3 (Goldstein & Uchida, 2016).

There is a notion of global anomalies and local anomalies in the literature. To explore this notion, Figure 2.9 illustrates this point. The data points x_1 and x_2 are clearly anomalies. Global anomaly detection techniques should be able to identify these point as such. However, the data point x_3 will cause an issue to global techniques and is likely to be mislabelled as a normal data point because it is close to the c_2 cluster. Thus, when approaching the anomalies in this sample dataset in a global manner, the algorithm is said to be a global anomaly detection algorithm. If the algorithm approaches each cluster individually, then it is likely to flag the data point x_3 as an anomaly and hence the algorithm will be referred to as a local anomaly detection algorithm. It is worth noting that c_3 lies in a grey area and could impose a challenge to anomaly detection algorithms. Should it be flagged as an anomaly or a normal cluster? This is part of the challenge that anomaly detection faces and usually, to determine to which class this cluster belongs, a domain human expert opinion is needed.

The unsupervised family of algorithms can be categorised as shown in Figure 2.10.

Nearest Neighbours		Clusters		Statistical	Spectral
Global	Local	Global	Local	HBOS	rPCA CMGOS
k-NN	LOF COF INFLO LoOP LOCI aLOCI	CBLOF uCBLOF	LDCOF CMGOS		

FIGURE 2.10: Categorisation of unsupervised anomaly detection algorithms.

2.4.3.1 k-Nearest Neighbour

The k-NN algorithm is a global anomaly detection algorithm and it could face problem detecting local anomalies such as the one illustrated in Figure 2.9. This group of algorithms can be further classified into two groups: k^{th}-nearest neighbours and k-nearest neighbours. The first group, k^{th}-nearest neighbours (Ramaswamy *et al.*, 2000), defines the anomaly score by calculating for each data point, the distance to the k^{th} nearest neighbour. The second group, k-nearest neighbours (Angiulli & Pizzuti, 2002), defines the anomaly score by calculating the average distance of the k-nearest neighbours.

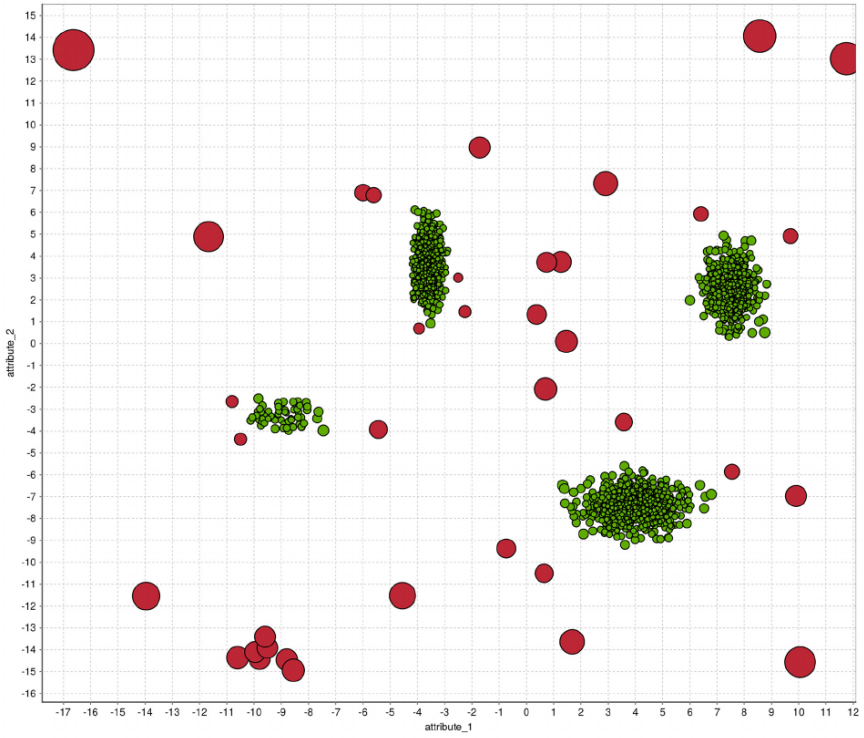


FIGURE 2.11: The k-nearest neighbour anomaly scoring of an artificial sample dataset (Goldstein & Uchida, 2016).

Figure 2.11 shows the result of applying the k-nearest neighbour algorithm on a sample artificial dataset. The red data points represent the anomalous data points and their radius corresponds to the anomaly score they got. In this algorithm, the k parameter must be set before running the algorithm. In this example $k = 10$. It can be seen how the algorithm assigns low anomaly scores for the data points closer to the green clusters.

2.4.3.2 Local Outlier Factor

The LOF (Breunig *et al.*, 2000) is one of the popular local anomaly detection techniques and many extensions and improvements have been proposed for it in the literature. The algorithm operation can be summarised into three steps:

1. Calculating the k-NN for every data point,
2. Calculating the local density based on the previous k-NN scores (N_k) by using the local reachability density (LRD) function for a data point x and an object o:

$$LRD_k(x) = 1/\left(\frac{\sum_{o \in N_k(x)} d_k(x,o)}{|N_k(x)|}\right) \tag{2.1}$$

   $d_k(x, o)$ is the reachability distance function,
3. Calculating the LOF score by comparing the LRD function of a data point with the LRD of its k-nearest neighbours.

$$LOF(x) = \frac{\sum_{o \in N_k(x)} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|} \tag{2.2}$$

In words, the LOF score is simply the ratio of the local densities. Therefore, normal data points will have densities similar to their local densities and the calculated anomaly score will be 1.0. The anomalous data points will get much larger score depending on how different is the data point density from its neighbours.

2.4.3.3 Connectivity-Based Outlier Factor

The COF (Tang *et al.*, 2002) algorithm is similar to LOF but it differs in the way the density is calculated. In LOF, the distances are Euclidean distances calculated using a hypersphere centred on a data point. Whereas COF calculates the distance in an incremental manner by finding the shortest paths between data points. This change in the way the distances are calculated was illustrated previously by Figure 2.8.

2.4.3.4 Influenced Outlierness

The INFLO is an extension to LOF proposed by Jin *et al.* (2006) to solve a shortcoming of LOF when there are two clusters of varying density close to each other. LOF mislabels the data points at the edges of the adjacent clusters. The INFLO overcomes this issue by incorporating a reverse nearest neighbours set of data points. To illustrate this idea, Figure 2.12 shows two clusters of varying densities. The red data point is flagged as an anomaly by LOF because in the hypersphere (the grey circle) there are 5 nearest neighbours which have high local density. INFLO will incorporate the reverse nearest neighbours set of data points (the blue data points) which will make it less likely for INFLO to consider the red point as an anomaly.

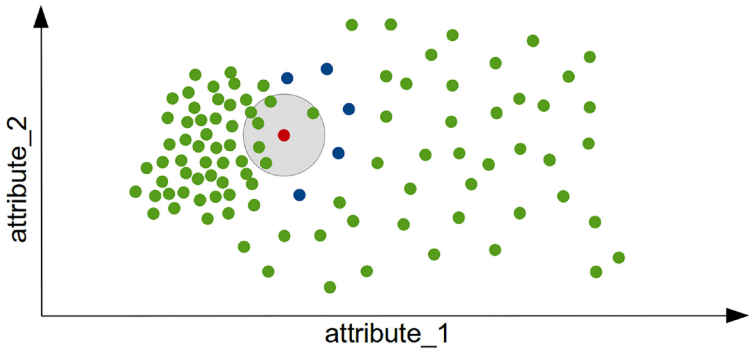


FIGURE 2.12: The INFLO algorithm compared to the LOF algorithm (Goldstein & Uchida, 2016).

2.4.3.5 Local Outlier Probability

The LoOP was proposed by Kriegel *et al.* (2009) to address the interpretation of anomaly scores of the previous algorithms. As explained earlier, some anomaly detection algorithms' output is binary which is limiting in some application. Other algorithms' output is a scalar value that measures how anomalous a data point is. Depending on the data points in the data set, this anomaly score can take arbitrary values which makes it hard to interpret the output of the algorithm. LoOP tries to mitigate this issue by producing a probability score of how anomalous a data point is.

2.4.3.6 Local Correlation Integral

The Local Correlation Integral algorithm (LOCI) is yet another improvement to the algorithms shown so far. The algorithm was developed by Papadimitriou *et al.* (2003). The main improvement that LOCI brings is providing a way to estimate a good value for

the crucial k parameter. The algorithm arrives at the best k score by iterating different values of k for each data point and the maximum score is taken for the corresponding k. This approach, however, is computationally expensive. Usually the k-NN approaches have $O(N^2)$ complexity whereas LOCI complexity can reach $O(N^3)$.

2.4.3.7 Approximate Local Correlation Integral

The approximate Local Correlation Integral algorithm (aLOCI) is an extension to LOCI to address the complexity issue. The algorithm speeds up LOCI operation by incorporating quad trees.

2.4.3.8 Cluster-Based Local Outlier Factor

The algorithms shown so far share in common their reliance on nearest-neighbours approaches. The Cluster-Based Local Outlier Factor algorithm (CBLOF) (He *et al.*, 2003) relies on using clustering approaches to identify anomalies. Any clustering algorithm can be used in conjunction with CBLOF as a first step. It is common to use k-means because of the low computational complexity it has. CBLOF then groups the clusters from the clustering algorithm into big and small clusters. The anomaly score is calculated as the distance of each data point to the cluster centroid times the number of data points in that cluster. An extension to CBLOF called unweighted CBLOF (uCBLOF) was proposed by Amer & Goldstein (2012) that excludes this scaling factor from the calculation as they noted that this factor could introduce issues when calculating the densities. Figure 2.13 shows the result of applying uCBLOF on a dataset. The different colours correspond to the clusters identified by the clustering algorithm used and the radius of the data points corresponds to the anomaly score assigned by uCBLOF to each data point.

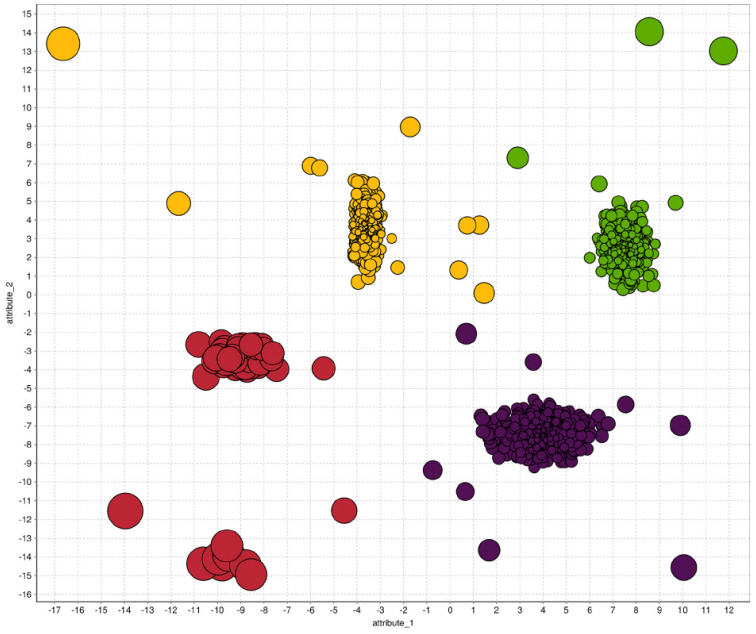


FIGURE 2.13: The unweighted Cluster-Based Local Outlier Factor (uCBLOF) algorithm (Goldstein & Uchida, 2016).

2.4.3.9 Local Density Cluster-Based Outlier Factor

One of the issues with CBLOF is its use of the number of a cluster data points as a density measure. The Local Density Cluster-Based Outlier Factor (LDCOF) (Amer & Goldstein, 2012) proposes a density measure of the identified clusters. It follows a similar approach to CBLOF by using any clustering algorithm as a first step. Then, the average distances from a cluster centroid to the data points that belongs to it are calculated. The anomaly score for LDCOF is calculated by dividing a data point distance to the cluster centroid by the cluster average.

2.4.3.10 Clustering-Based Multivariate Gaussian Outlier Score

As its name suggests, the Clustering-Based Multivariate Gaussian Outlier Score (CMGOS) (Goldstein & Uchida, 2016) depends on using a clustering algorithm as a first step. CMGOS the cluster density is calculated using a multivariate Gaussian model with Mahalanobis distance as the measurement function. After identifying the clusters by a clustering algorithm, a covariance matrix for each cluster is calculated. The anomaly score is then defined by dividing the Mahalanobis distance of a data point by the χ^2 distribution of the confidence interval.

2.4.3.11 Histogram-Based Outlier Score

The Histogram-Based Outlier Score algorithm (HBOS) (Goldstein & Dengel, 2012) is a statistical anomaly detection algorithm that assumes that the feature space is independent. The algorithm's main idea is to build a histogram for every variable (feature or dimension) in the dataset. For each data point, the height of the bin represents a density estimator. The final score is the multiplication of the inverse of estimated densities. Although the assumption that the features are independent is limiting, this assumption gives HBOS an advantage when dealing with high dimensional dataset as the algorithm complexity is linear in relation to the input size.

2.4.4 Anomaly Detection Applications and Domains

Anomaly detection can play important roles for different applications. Various anomaly and outlier detection methods can be applied on datasets as a pre-processing step to prepare the data. Other techniques can be used to build models that are able to detect anomalies in different scenarios. The domains that anomaly detection can be applicable to are broad. In this Section, several domains and applications of anomaly detection will be presented.

2.4.4.1 Intrusion Detection

Intrusion detection is a form of abusing a computer system by an unauthorised access for malicious intents. Detecting these malicious activities is of a particular interest to computer security experts. Different anomaly detection techniques can be used to identify these anomalies. One of the major issues in this domain is the sheer amount of data that needs to be inspected. Another issue is the need to detect these anomalies as soon as possible. Therefore, online detection techniques are important here. The anomaly detection models in this domain can learn the normal behaviour of a computer system since the data of the normal operation of the computer system are available. Thus, the models in this domain generally learn in a semi-supervised approach (Chandola *et al.*, 2009).

This domain can be further categorised into two sub-categories, **host** and **network intrusion**. The anomaly detection in the first category tries to identify when an adversary tries to gain access to a host computer system without permission. The data usually form a sequence of system calls and the anomaly detection techniques task is to detect any abnormal sequence of system calls. Examples of this approaches is what Cabrera

et al. (2001) has done by using sequences of Unix system calls generated by various programs to classify a given activities as normal or abnormal. Eskinand & Stolfo (2001) modelled the system calls using dynamic window sizes. Marceau (2001) used N-grams representing the strings of the system calls to detect anomalous activities. Heller *et al.* (2003) used One-Class Support Vector Machine (OCSVM) to classify anomalous access to Microsoft's Windows registry. Hu *et al.* (2003) used Robust Support Vector Machines (RSVMs) and compared their performance against traditional Support Vector Machines (SVM) on the 1998 DARPA intrusion dataset Kendall (1999).

For network intrusion, generally the anomalies form point anomalies [2.4.1] but there are techniques in the literature that model the anomalies as collective anomalies [2.4.1] similar to (Gwadera *et al.*, 2005; Atallah *et al.*, 2004). In this category, outside intruders try to gain access to a computer network. The available data has a temporal dimension and usually is in the form of network packets and other network metrics. These metrics can be high-dimensional and of various numerical and categorical types. Kruegel & Vigna (2003) developed an intrusion detection for web servers using different statistical techniques. Chan *et al.* (2003) proposed a clustering algorithm called Clustering for Anomaly Detection (CLAD) to identify anomalies. Yeung & Chow (2002) used a density based approach with Parzen-window estimators and Gaussian kernels to detect anomalies in computer networks. Sun *et al.* (2007) proposed a dimensionality reduction technique called Compact Matrix Decomposition (CMD) to lower the dimensionality of the data in computer networks. Ramadas *et al.* (2003) used Self-Organising Maps (SOMs) to detect anomalies in network traffic.

2.4.4.2 Fraud Detection

In this domain, anomaly detection techniques are used to detect fraudulent transactions for credit cards, banks, commercial companies. Criminal activities such as identity theft can also use forms of anomaly detection techniques to identify the criminals. A profile of the normal behaviour of a customer is maintained by a Bank, for example, and when abnormal activities are detected such as withdrawals from unusual locations, the anomaly detection technique sends an alert.

Detecting fraud for credit cards and insurance is similar in that the data is usually in the form of a user ID and their transactions are measured with different metrics. The anomaly type can be classified as point anomaly [2.4.1] because when a fraudulent transaction happens, it usually has a different characteristics from the normal transactions performed by the customer. The difference could be the time when this transaction occurred, the location, or the quantity. One of the main challenges in this domain

is having an anomaly detection technique that is able to quickly detect the abnormal behaviour. Therefore, online techniques are preferred.

Aleskerov *et al.* (1997) used an auto-associative neural network model to develop a technique called CARDWATCH to detect fraudulent credit card transactions. Similarly, Brause *et al.* (1999) used neural networks to detect anomalies with low rate of false alarms. Bolton *et al.* (2001) applied unsupervised fraud detection based on clustering techniques on several credit card datasets. Phua *et al.* (2004) developed a method that uses a back-propagation algorithm with a Naive Bayes model to detect fraud on an automobile insurance company dataset. Brockett *et al.* (1998) used Kohonen's Self-Organising Feature Maps to classify fraudulent automobile bodily injury claims.

2.4.4.3 Health and Medical

Public health and medicine can make use of several anomaly detection techniques for several purposes such as detecting recording mistakes. The data in this domain is the patient's records with many different types such as the patient's name, age, weight, blood type, condition, etc. The data have spatial and temporal characteristics. Most of the anomaly techniques in this domain deal with point anomalies [2.4.1] and semi-supervised approaches are used because of the availability of healthy patients records. The temporal aspect can play an important role when dealing with Electrocardiograms (ECG) data and Electroencephalograms (EEG). There is also collective anomaly detection research such as the work done by (Lin *et al.*, 2005).

A mistake in identifying an anomaly in this domain could have dire consequences because of the sensitivity of the subject. Wong *et al.* (2003) used Bayesian networks to detect outbreaks of diseases. Solberg & Lahti (2005) used statistical techniques to detect anomalies in medical laboratory reference data as a pre-processing step. Suzuki *et al.* (2003) applied probabilistic mixture model to visualise outliers in medical test data.

2.4.4.4 Image and Video

In this domain, anomaly detection techniques are used to detect changes in still images, stream of images and video clips. Several sub-domains can be categorised under this domain such as video surveillance, spectroscopy, hand writing recognition, satellite imagery, audio analysis etc. One of the biggest challenges in this domain is the high dimensionality and the sheer number of data points. For example, an image is composed of pixels and each pixel can be described by three colour components (red, green, blue).

In 3D imaging, other information is associated with the data points such as texture and luminosity.

Pokrajac *et al.* (2007) proposed an extension to the Local Outlier Factor (LOF) algorithm called incremental LOF to detect anomalies in data streams. The technique was evaluated on a dataset of video clips. Singh & Markou (2004) proposed a framework that uses neural networks as classifiers to classify anomalous regions of images. Davy & Godsill (2002) proposed a machine learning algorithm based on SVMs to find sudden changes in audio streams. Da *et al.* (2005) proposed a regression model to detect anomalies in multivariate near-infrared spectroscopic data sets.

2.4.4.5 Textual Data

Anomaly detection can be used in this domain to detect emerging stories and news. For example, analysing Twitter traffic to identify breaking news. The data here usually has high dimensional features and has temporal aspects. Srivastava & Zane-Ulman (2005) detected anomalies in large textual datasets by applying a clustering technique based on Von Mises-Fisher distribution and compared their techniques with two techniques based on k-means and Gaussian mixture models. Miller *et al.* (2014) applied clustering techniques to detect spam in Twitter traffic. Manevitz & Yousef (2001) used One-Class Support Vector Machine (OCSVM) to detect anomalies in the Lewis (1997) dataset.

2.4.4.6 Wireless Sensor Networks

Anomaly detection in the Wireless Sensor Networks (WSNs) domain uses readings from sensors distributed across a network to detect intrusion or identify faulty sensors. The types of data in this domain can be in several numerical discrete or continuous form, categorical, video, audio, etc. The data suffer from a high degree of noise and missing data points due to sensors failure, environmental conditions and communication means issues. One of the requirements for most of the anomaly detection techniques in this domain is to operate in an online fashion. Moreover, the data is collected from distributed sources which requires approaches applicable to their nature (Chatzigiannakis *et al.*, 2006). Also, the data contains a fair amount of noise which makes anomaly detection more difficult.

Janakiram *et al.* (2006) used Bayesian Belief Networks (BBNs) to detect spatial and temporal anomalies in the sensors streaming data. Van Phuong *et al.* (2006) proposed a statistical anomaly detection algorithm to detect security attacks in WSNs. Branch *et al.* (2013) proposed a rule-based algorithm to detect anomalies in WSNs and validated

the results using SENSE wireless sensor network simulator (Chen *et al.*, 2005b). Idé *et al.* (2007) proposed a technique based on nearest neighbours to detect changes in correlated streams of sensors.

2.5 Hierarchical Temporal Memory

The network traffic generated online nowadays is huge and gigantic. Chen (2012) reported that the estimation of the Internet traffic produced by 20 houses in the year 2012 will be more than the whole Internet traffic in the year 2008. Moreover, in 2003 the world population was 6.3 billion people and the number of online connected devices was 500 million and in 2010, the world population was 6.8 billion and the number of connected devices was 12.5 billion which meant that the number of devices per person is more than 1 for the first time in history (Evans, 2011).

Taking the human ear and how it receives sound, for example. There are around 30 thousand receptors (sensors) in the cochlea that receive sound vibration and send these vibrations up the hierarchy for further processing. So the brain is capable of processing this huge number of ever changing input streams coming via thousands of sensors with ease.

In the IoT paradigm, the number of sensors and data generated will be huge and there is a biological machine that is capable of dealing with such number which is the human brain. HTM takes the brain as its source of inspiration and design guidelines, using an algorithmic implementation called the CLA.

2.5.1 Anomaly Detection Using CLA

Anomaly detection in HTM is unsupervised and is targeted at streaming or time series data. There is a commercial product for IT analysis called Grok[2] built from the same open source code base of NuPIC. Grok for IT is one of several products that Numenta created. The interesting thing about these products is that they all share the same algorithm and the same code base even though the applications of these products are in different domains. Here is a list of all the products that Numenta produces:

- **HTM for stocks**: detects anomalies in stock market.
- **Rogue behaviour detection**: detects abnormal behaviour of individuals in a company such as accessing an unauthorised file or abnormal downloading activities.

[2]http://numenta.com/grok/

- **Geo-spacial tracking**: detects anomalies in geo-spacial data.

Table 2.1 shows a comparison of different anomaly detection techniques used in unsupervised streaming data. The numbers (1, 2, 3, 4) represent different anomaly types and how well a given technique will detect them (Hawkins, 2014). Here is an explanation of the types of anomalies that Grok can detect:

- **Anomalies of type 1:** Easily detectable anomalies due to huge shift in the data.
- **Anomalies of type 2:** Subtle anomalies in periodic data.
- **Anomalies of type 3:** Anomalies in highly noisy data.
- **Anomalies of type 4:** Anomalies that a human monitor will likely not notice.

TABLE 2.1: Comparing Grok against other anomaly detection methods (Hawkins, 2014).

Technique	**1**	**2**	**3**	**4**
Simple Threshold	Yes	No	No	No
Complex Statistical	Yes	Maybe	Yes	No
Time Series Analysis	Yes	Yes	No	No
Distance based	Yes	Maybe	No	No
Supervised Methods	N/A	N/A	N/A	N/A
Grok (HTM based)	Yes	Yes	Yes	Yes

The simple threshold techniques use simple statistical models and requires manual configurations and maintenance. Some of the more complex statistical techniques can catch the anomalies of type 2. Time series techniques such as Holt-Winters, Autoregressive Integrated Moving Average (ARIMA), Support Vector Regression (SVR) can detect anomalies of type 2 if the period of the time series is manually set. Distance based techniques are unsupervised techniques. On the other hand, the supervised techniques such as Replicator Neural Networks (RNNs), Bayesian Networks, Rule-based, Support Vector Machines (SVMs) rely on the availability of training data with accurate labels which is hard to do in an online, streaming fashion. On the other hand, the HTM based and commercial product Grok was able to detect the anomalies of type 3 and 4 (Hawkins, 2014).

2.5.2 Numenta Platform for Intelligent Computing (NuPIC)

NuPIC is an open source project that implements the CLA based on the 'memory-prediction' or the HTM theory of Jeff Hawkins.

2.5.2.1 NuPIC Advantages

- **Online Learning**

In most machine learning work flows, there is a training dataset that is used to train the model and then the performance of the model is evaluated against another testing dataset. After that, the model can be applied on new datasets. NuPIC does not follow this work flow, the model is constantly *learning* and producing results. The accuracy of the model improves over time and produces better results. This flexibility is needed for a smart home environment that will receives a huge number of data streams from different smart devices. Also, since NuPIC is a memory system, there is no need to store all of the data streams to be able to perform predictions and anomaly detection. The data streams come as an input to the model to shape and morph the model over time.

- **Noise tolerance**

NuPIC uses Sparse Distributed Representations (SDRs) to encode the data. SDRs are very resilient against noisy data and the nature of the data in a smart home setting is very noisy.

- **Predictions and anomaly detection**

NuPIC is able to predict values that are multiple time steps in the future. In addition to the prediction capabilities, NuPIC scores an anomaly detection value for each incoming input record.

- **Human-like intelligence**

In one of the introductory examples on the NuPIC web site[3], there is a sample experiment for predicting the values of a sine wave. This example is interesting because it reveals how the implementation of NuPIC is faithful to how our intelligence is. Figure 2.14 shows a snapshot of the performance of NuPIC predicting the sine wave. The

[3]https://github.com/subutai/nupic.subutai/tree/master/swarm_examples

actual sine plot is coloured blue and the value that NuPIC is predicting is coloured red. At the beginning, NuPIC will just predict the same value that it just received. This is why a trailing behaviour at the beginning is observed. Looking at Figure 2.15 which shows the anomaly score that NuPIC assigns to every prediction value, at the beginning stages the anomaly score is high but gradually the score is lowered for each cycle of the sine wave.

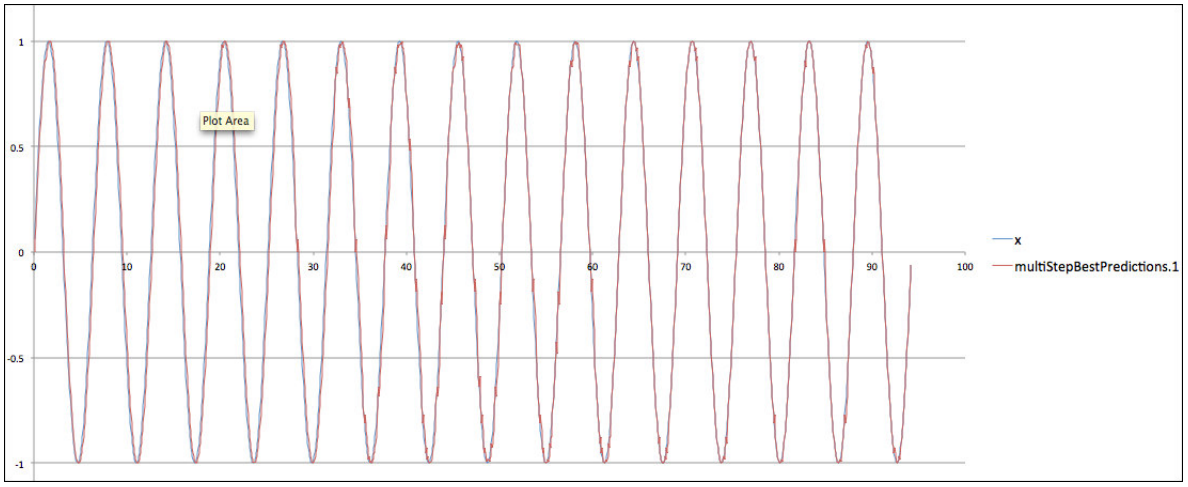


FIGURE 2.14: NuPIC predicting the sine wave.

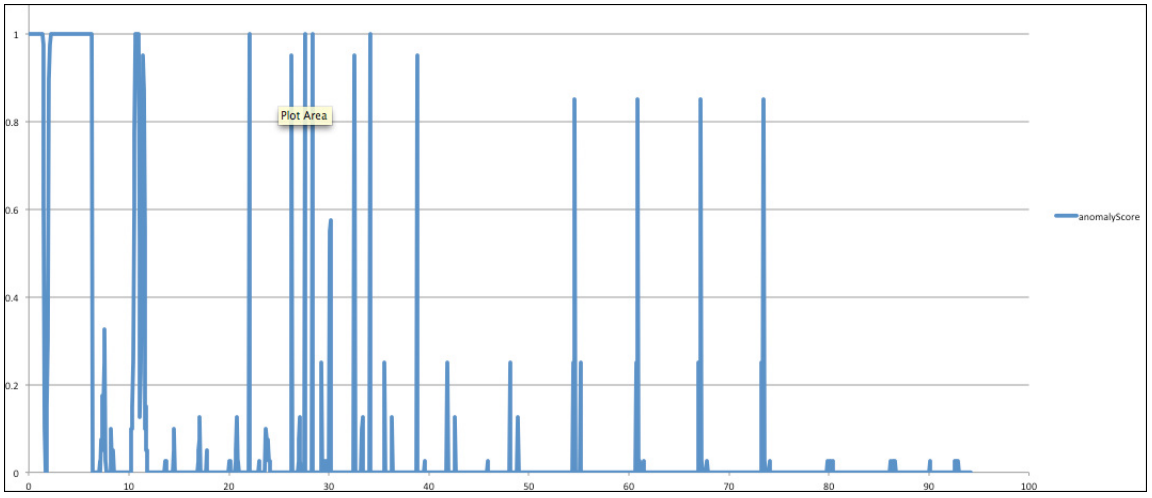


FIGURE 2.15: NuPIC anomaly score.

The interesting thing about this example is how NuPIC predicts the wrong values. Figure 2.16 shows that even after some time NuPIC cannot predict the values exactly even though the pattern of the data is perfect and cyclic. These spikes in the predictions manifest the human-like intelligence that NuPIC possess. If one tries to teach a youngster how the sine wave works by giving him pairs of values that represent the input and output values of the sine wave and then the teaching process will be based completely on remembering that when he/she is asked about a certain input value, the answer will be the other output value. The youngster undoubtedly will make wrong answers because

it is hard to remember all of these pairs. It will take a lot more time to learn how the sine wave works by following this method compared to mathematically explaining how the sine wave function works. It turns out that NuPIC is not the best technique to use if the nature of the data can be explained easily by a mathematical equation.

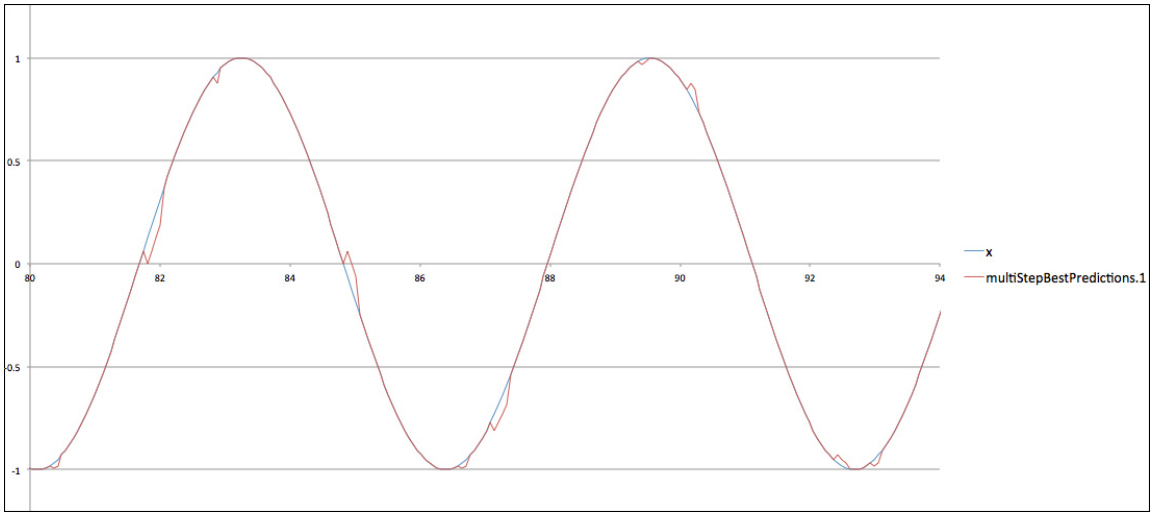


FIGURE 2.16: NuPIC small errors in predicting the sine wave.

The nature of the data generated in a smart home environment is derived from the human user behaviour. This human-like intelligence that NuPIC has might help in producing better anomaly detection results than other machine learning techniques.

- **Generalisability**

Having a good solution for one smart home setting, will allow us to generalise the solution for all homes. This is due to the way the system functions, and how it is not a domain specific learning algorithm.

2.6 Smart Homes

Smart home research can be thought of as a subset of ubiquitous computing research. A smart home should exhibit forms of intelligent behaviours and reactions to the home inhabitants. The areas in which these forms of intelligence take place can be safety, healthcare, privacy, security, energy consumption, entertainment and comfort. Compared to traditional homes, a smart home provides its inhabitants with better living standards. Learning from the inhabitants' habits and behaviours allows the smart home to automate repetitive tasks, ease control over the home, and provide assistive smart services.

2.6.1 Definition

Lutolf (1992, pg. 277) defined a smart home as:

> "The smart home concept is the integration of different services within a home by 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."

The previous definition comes from a home automation background and does not convey the importance of intelligence and context-awareness.

Another definition of the concept of smart home can be found in the work of van Berlo *et al.* (1999, pg. 4) who defined it 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 is broad and generic and can be applicable to smart homes and other forms of smart environments.

Briere *et al.* (2011, pg. 16) defined the smart home as:

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

This definition is also broad and does not establish what a smart home is.

A more refined definition is what Intertek (Alam *et al.*, 2012, pg. 1191) published in 2003 and defined a smart home as:

> "A smart home is a dwelling incorporating a communication network that connects key electrical appliances and services and allows them to be remotely controlled, monitored, or accessed."

According to Intertek, the smart home concept needs three components:

- Network

- Intelligent control
- Home automation

The network component can be of any type whether wired or wireless. The intelligent control is a gateway or a dashboard that allows the home inhabitant to control and manage the home. The home automation is the capability for the smart home to connect and work with services outside the premise of the home.

Satpathy (2006, pg. 43) defined the smart home concept as:

> "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."

A more comprehensive definition is proposed by Alam *et al.* (2012, pg. 1191):

> "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."

2.6.2 Applications and Projects

There are many applications and uses for smart homes and these applications are evolving and emerging as new technologies and advancements are taking place. The IoT paradigm is one of the recent advancements that pushed the capabilities and applications of the smart home. Though these applications are diverse and different, Alam *et al.* (2012) proposed categorisation of the smart home applications according to the intended services:

- Comfort
- Healthcare
- Security

The inhabitants comfort entails recognising and identifying the inhabitants activities and automating repetitive tasks. Remotely accessing the home and controlling it is also another example of smart home applications that increase the comfort and quality of life for the inhabitants.

Healthcare care is one of the most prominent applications of smart home especially for the elderly as life expectancy is increasing and many elderly prefer living independently in their homes. Healthcare can be local for the inhabitants in the home or remote by allowing trained personnel to monitor and observe the activities of the elderly.

Security is another important application of the smart home especially as the vision of the IoT is becoming a reality. It is vital for the home inhabitant to be able to authenticate and securely access and control his/her smart home.

2.6.3 Anomaly Detection in Smart Homes

Anomaly detection has many applications and many research interests in the smart home field. One of these research interest is health monitoring for the elderly and assisted independent living. Using the MavHome project (Cook *et al.*, 2003a), Jain *et al.* (2006) generated a week worth of inhabitants' data. The generated data was 1400 events per day of the week. Part of that work was to detect anomalies in the data and they used a simple statistical anomaly detection technique (z-scores) that flags an event as an anomaly if its value is extremely high or low compared to whole data.

Novak *et al.* (2012) proposed an unobtrusive anomaly detection technique for the elderly in a smart home environment. The authors used the MavHome project dataset and built a clustering anomaly detection algorithm using Self Organising Maps (SOM), which are a type of artificial neural networks, that detect several types of anomalies such as unusual long or short periods of inactivity, unusual presence or absence and changes in the daily rhythms.

Shin *et al.* (2011) proposed a method for detecting anomalous living patterns for the elderly who are living alone. The anomaly detection technique used a Support Vector Data Description (SVDD) method to classify a given pattern as normal or abnormal. The dataset for this project collected infrared (IR) motion sensors installed in elderly inhabitants homes. The test subjects were suffering from mild to sever diseases. The data collection lasted for seven months.

Another research interest is monitoring power consumption for smart homes and detecting anomalies when they occur. Jakkula & Cook (2010) used the CASAS smart environment project Cook *et al.* (2009) to generate the power consumption dataset along with a

synthetic dataset. The real dataset contained three months worth of data. The synthetic dataset contained twelve months worth of data. The authors then injected anomalies in the datasets. The anomaly detection technique in this research was a simple statistical algorithm that uses the t-score to flag an extreme event value as an anomaly. Later on, the authors used a clustering approach with k-Nearest Neighbour (k-NN) algorithms which gave better accuracy in detecting anomalies over their real and synthetic datasets.

Kang *et al.* (2010) used a Hierarchical Hidden Markov Model (HHMM) to recognise and predict the states of a smart home inhabitant and proposed an algorithm based on that model to detect anomalies in the inhabitants behaviour. 77 sensors were installed in two single-person apartments and gathered data for two weeks. The sensors were installed on devices such as refrigerators, drawers, etc. Inspired by the work of Intille *et al.* (2003), an Experience Sampling Method (ESM) were used to label the activities and the patterns.

2.6.4 Requirements for Anomaly Detection in Smart Homes

For an intelligent system to be able to detect anomalies in smart homes, it needs to have certain qualities that enable it to cope with the nature of the data in this domain. One of the main requirements is the ability for the system to operate in unsupervised fashion without the interference of a human agent to train or tweak the systems parameters. This requirement is essential since it is hard to define what an anomaly is for a smart home inhabitant. Each person possess unique daily habits and therefore, the notation of anomalous behaviour is very subjective construct. Thus, the unsupervised techniques are suitable to detect anomalies in a smart home setting.

Another important characteristic of an anomaly detection system for smart homes is the ability to work in online fashion. Late detection of anomalies in most cases is not beneficial, the system should be able to detect anomalies as early as possible. The streaming nature of the data in smart homes, requires the system to be able to cope with the volume of the incoming data. The system should also be fast enough to produce results in close to real-time as possible.

From the perspective of the smart home sensors, the anomalies in smart home could be spatial. Meaning an unusual configuration of activities in the sensors' readings should signal an anomaly. Moreover, temporal anomalies are also important anomalies to be detected. For example, if a smart home has several Passive Infrared (PIR) sensors in the living room, it is not usual to have them active at the same time when someone is at the living room (spatial configuration). However, if the same sensors are active at 3:00

am, this could be a security issue. So, the time dimension should be taken into account when detecting anomalies.

2.6.5 Intelligent Services in Smart Homes

Learning the habits of a smart home user is an important and difficult task to be achieved. The survey by Soma Bandyopadhyay *et al.* (2011), shows that most of the middleware projects focus on device management and neglect context-awareness and interoperability. After reviewing and analysing 50 IoT and smart home middlewares, Perera *et al.* (2014) concluded that the context-awareness aspect has not been fully addressed. Machine Learning could have the potential to achieve good context-aware services provided by the middlewares. By incorporating multiple Machine Learning models, sophisticated assemblies of these models can open the doors to many innovations and intelligent services.

An example of these Machine Learning intelligent services is what have been done by Google's Android mobile system to provide an intelligent service to locate a parking car. When one drives a car and parks at a certain location, a mixture of multiple Machine Learning models work in tandem to provide a useful service to the user. There is a classification model that can identify whether the user is walking, running, cycling, riding a train, or driving a car. When the user transitions from 'driving' a car to 'walking', the system infers that the user had parked his car. The system then takes note of the car parking location using the GPS enabled phone. When the user walks towards the direction of the parking space, the phone intelligently and contextually shows a notification to the user of where his car is located.

For the smart home, an anomaly detection system could be a valuable Machine Learning model that can be used in many creative and context-aware intelligent services for many applications.

2.7 Early Experimental Results

Since the HTM theory and its algorithmic implementation, the CLA, have not been tested for anomaly detection in a setting where many sensors generating streaming data constantly, a quick experimental study have been conducted to evaluate how good the results will be. A prototype simulation tool has been built (which later on became OpenSHS, see Chapter 4) to generate a dataset of an inhabitant living in a smart home with

multiple sensors. Ten artificial anomalies have been injected into the dataset and the performance of the CLA and DBSCAN have been evaluated.

2.7.1 Dataset

The researcher performed simulations using a simulation tool of typical daily habits of a smart home inhabitant. All of the simulations were made manually to ensure a level of realism of the recorded data. Several considerations were made while creating the dataset:

- All of the simulations took place roughly at the same time of the day.
- A small time period at the beginning of the home inhabitant's day was the target time for this dataset. The interest was on the early morning habits of the inhabitant.
- Since the simulations were done in real-time, similar patterns are not identical and varied from day to day.
- There were two distinct patterns, one for the weekdays and the other for the weekends.
- The actions performed varied from day to day. For example, sometimes the bed room lights are turned on when the avatar wakes up and sometimes they are left off.

The early experimental dataset is 4400 records long and captured 3 months worth of data of that early morning period of the inhabitant's habits. Figure 2.17 shows a sample of the dataset records and columns.

bathroomDoor	bedTableLamp	bedroomDoor	bedroomLight	hallwayLight	laundryDoor	livingLight	mainDoor	officeDoor	PowerConsumption	room	tv	timestamp
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:08
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:09
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:10
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:11
0	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:12
0	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:13
0	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:14
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:15
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:16
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:17
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:18
1	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:19
0	1	1	1	0	0	0	0	0	6.0	1	0	2015-10-22 15:37:20
0	1	1	1	0	0	0	0	0	6.0	2	0	2015-10-22 15:37:21
0	1	1	1	0	0	0	0	0	6.0	2	0	2015-10-22 15:37:22
0	1	1	1	0	0	0	0	0	6.0	2	0	2015-10-22 15:37:23

FIGURE 2.17: A sample of the dataset.

2.7.2 Preparation for the CLA

Before feeding the data into the CLA, a process called **swarming** must be performed to tune the parameters of the model on the data set. The swarm process evaluates many models and chooses the best one according to an error metric. One thing to note is that the swarming process is optimised for classification problems. More work is needed to make the swarm optimised for anomaly detection problems.

After running the swarm process, the best model parameters and encoders are identified and a model was built using these parameters. Then, the dataset records were fed to the model record by record to allow the model to learn from the data. After a certain amount, the learning was stopped and the model was put in a testing phase. Figure 2.18 shows the anomaly likelihood of the dataset. The anomaly likelihood is a percentage score from 0% to 100% where zero means there is no anomalies detected and 100% means an anomaly happened with high certainty.

2.7.3 CLA Results

As shown by Figure 2.18, the anomaly likelihood starts at 50% and stays at that level until sufficient data is fed. The anomaly likelihood starts decreasing once it learns the habits of the smart home user. After the 3400th record, the training was stopped and the evaluation process started. There were 10 anomalies introduced in the remaining dataset records that will be discussed in the following Sections.

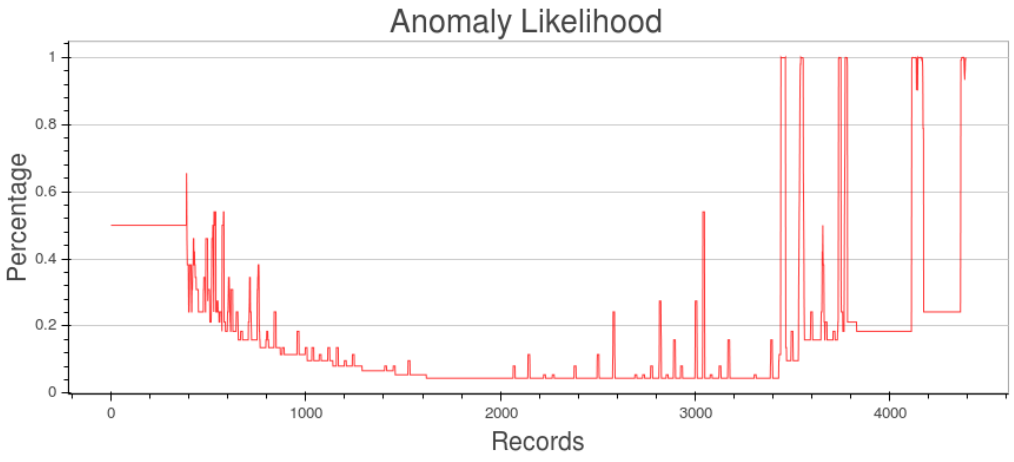


FIGURE 2.18: Anomaly likelihood scores.

2.7.4 DBSCAN Algorithm

The DBSCAN algorithm (Ester *et al.*, 1996) is a famous clustering algorithm and cited a lot in the literature. In 2014, the DBSCAN won the **Test of Time Award** from the prestigious KDD conference [4]. In this experiment, this algorithm was chosen to compare it against the performance of the CLA. The BDSCAN algorithm was chosen because it is a well-known clustering algorithm and operates in an unsupervised fashion to detect anomalies. Scikit-learn[5] was used, which is a Python library used in machine learning research and development. Figure 2.19 shows some of the clustering algorithms available in the library[6] and also shows a comparison between all of these algorithms on some samples. The DBSCAN correctly clustered the data points and outperformed the other algorithms on the Scikit-learn sample datasets. In the Scikit-learn implementation of the algorithm, when feeding a record to the algorithm, it will output a zero or positive number representing the cluster label that the record belongs to. If the record does not belong to any cluster, it will consider it an outlier and outputs the label -1.

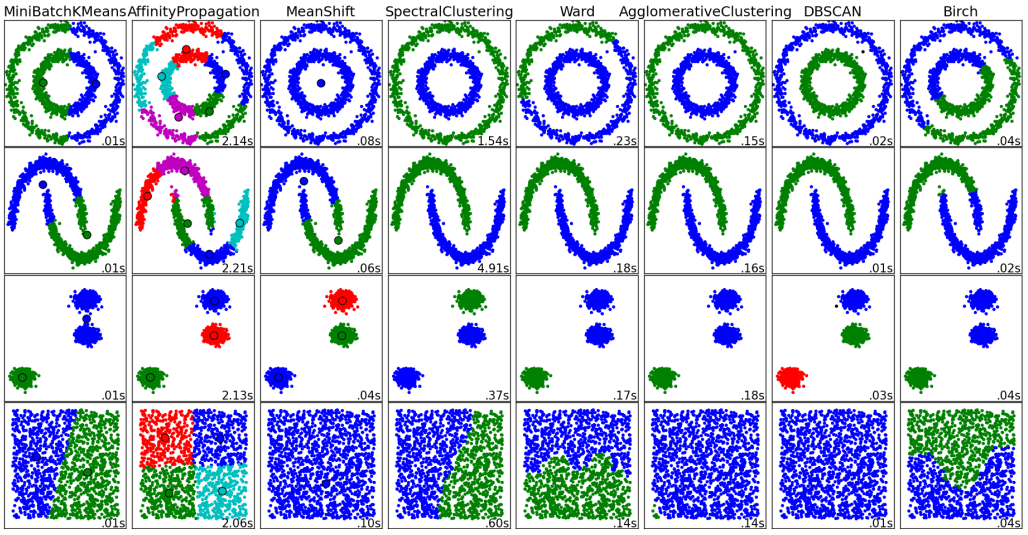


FIGURE 2.19: Overview of Scikit-learn clustering algorithms.

Figure 2.20 shows the results of feeding the dataset to the DBSCAN algorithm. All of the data points below 0 are considered anomalies.

[4] http://kdd.org/awards/view/2014-sikdd-test-of-time-award-winners
[5] http://scikit-learn.org/
[6] http://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods

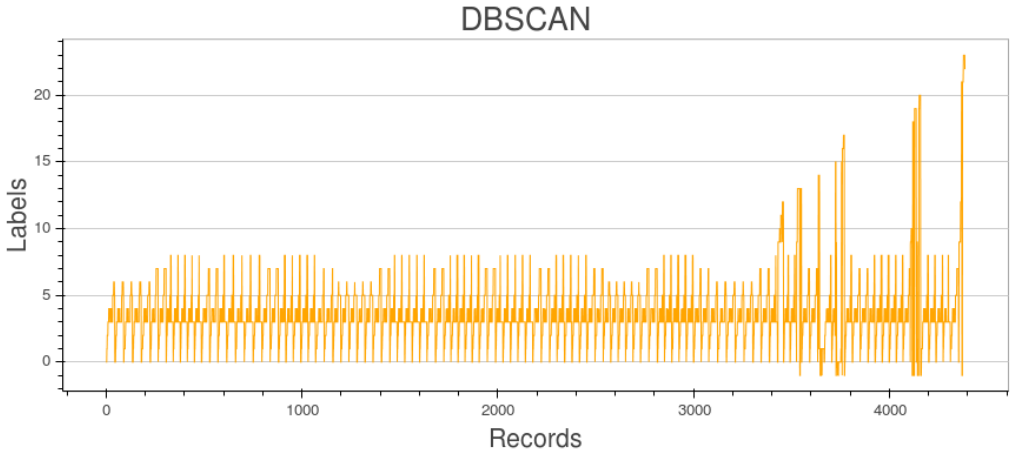


FIGURE 2.20: DBSCAN results on the dataset.

2.7.5 Anomalies in the Dataset

The simulation tool was used to introduce ten anomalies in the later part of the dataset (from the 3400th record onward). Some of the anomalies were strong and obvious and some were subtle. For example, one of the strong anomalies is leaving the main door open when leaving the house. Another example of the subtle anomalies is doing a usual pattern such as waking up in the morning and going to the bathroom and then going to watch TV but in a different context. That pattern usually happens in weekends and usually the user do not watch TV in the morning because he leaves for work.

Because the focus was on a small period of time (the early morning habits) for each day in the dataset, the whole period was annotated as an anomaly to simplify the evaluation process. In the future when bigger dataset are used, the anomaly annotation will be fine grained to each individual pattern.

2.7.6 Comparing CLA with DBSCAN

Figure 2.21 shows both of the algorithms on the same dataset. It is worth noting that the CLA outputs an anomaly percentage and the DBSCAN outputs a -1 label for the anomalies. Because of the different outputs, the CLA output was thresholded to match that of the DBSCAN. The highest number the DBSCAN outputted was 23.

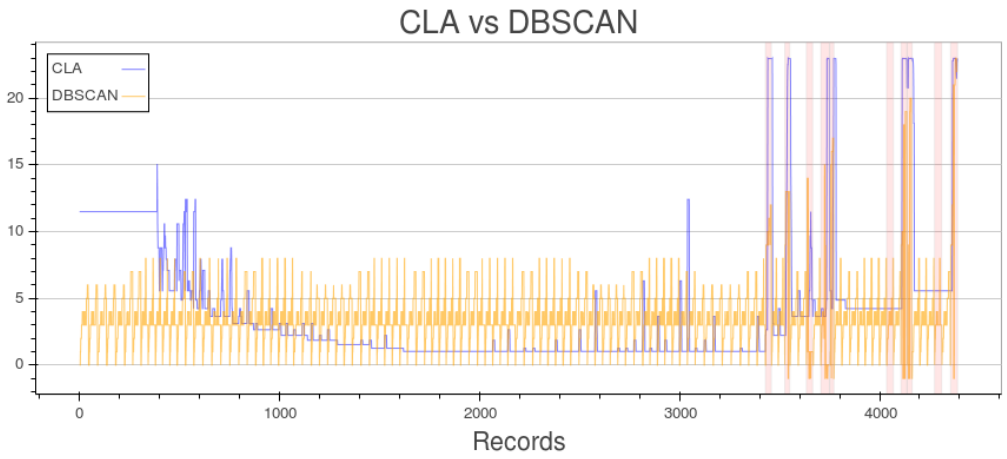


FIGURE 2.21: The CLA and DBSCAN results combined.

Figure 2.22 shows a zoomed view on the latter part of the dataset where the anomalies are introduced. The anomalous days are highlighted in red which are 10 anomalies. Some of them are adjacent to each other and that is why the highlighting seems wide in these instances.

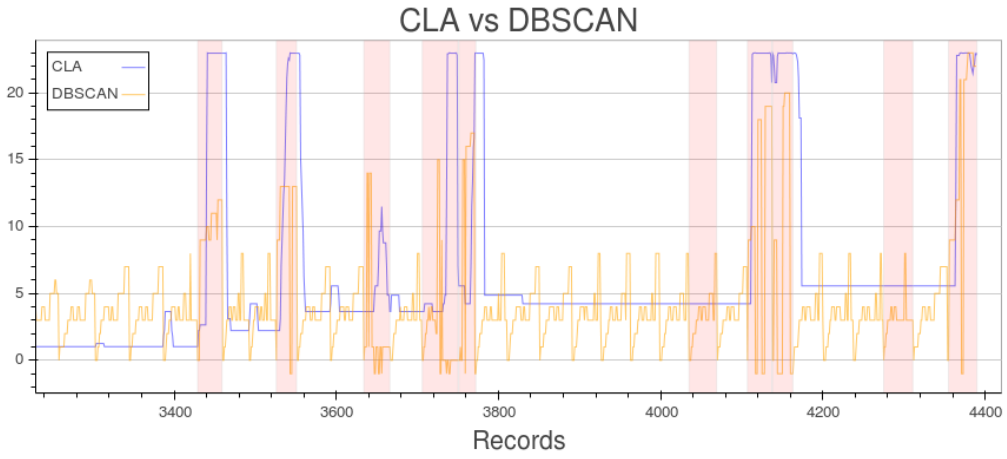


FIGURE 2.22: The CLA and DBSCAN anomaly results.

2.7.7 Discussion

When considering the case of detecting at least a single anomaly within the anomalous period, and a threshold value greater than 99% for the CLA algorithm, as our metric for counting correctly identified anomalies, both of the algorithms correctly detected 7 out of the 10 anomalies. Table 2.2 shows a summary of the results. In the anomaly number 3, the CLA had a spike in the anomaly score and the value was 50%. If the threshold of what is considered an anomaly was lower than 50%, the CLA will score 8 out of 10 in this small experiment.

TABLE 2.2: CLA vs DBSCAN.

Anomaly	CLA	DBSCAN	Notes
1	Detected	Not Detected	Strong anomaly
2	Detected	Detected	Strong anomaly
3	Not Detected	Detected	Subtle anomaly, CLA anomaly score is 50%
4	Detected	Detected	Strong anomaly
5	Detected	Detected	Strong anomaly
6	Not Detected	Not Detected	Subtle anomaly
7	Detected	Detected	Strong anomaly
8	Detected	Detected	Strong anomaly
9	Not Detected	Not Detected	Subtle anomaly
10	Detected	Detected	Strong anomaly

The CLA results seem promising even though the swarm process had not being optimised for anomaly detection. Also, the dataset used here is small and a bigger dataset that spans longer periods of time will be built by the simulation software.

2.8 Research Gaps

Anomaly detection research in the context of smart homes had been studied in the literature but more research efforts are needed. The anomaly detection challenges identified at the beginning of this Chapter (Section 2.4.1) do apply to the smart home domain. When taking these challenges into account and analysing the currently available literature, the lack of well-studied anomaly detection in the smart home domain becomes apparent.

Novak *et al.* (2012) used Self Organising Maps (SOM) to detect anomalies in a smart home. The aim of the authors' study was to develop an unobtrusive anomaly detection technique based on the presence of the inhabitants. They used a dataset from the MavHome project (Cook *et al.*, 2003a) and made modification to the dataset to suit their needs. For each room in the dataset, the sensors' readings were merged into one logical sensor reading. This preprocessing step made it simpler to feed the data to the machine learning model. Although this simplification is suitable for their aim, it can potentially lead to over-simplification and loss of valuable information that can be learned from the

data. The authors created artificial anomalies and injected them in the dataset. They defined anomalies as:

- Unusual long inactivity,
- Unusual short activity,
- Unusual presence.

An issue arises here, how to decide what *unusual* really means. The authors decided to have fixed thresholds for what is unusually short or long activity and inactivity respectively. However, the anomalies are highly contextual and subjective to the inhabitants' patterns.

The lack of real-world smart home datasets targeted at anomaly detection problems specifically is another gap in the literature. Usually, smart home datasets that are generated for classification problems is used. An augmented notion of anomalies is introduced to the datasets after the fact. These introduced anomalies could be manually injected and annotated in the datasets by the researcher Jakkula & Cook (2008), or introduced using a definition of anomalies produced by the researcher and not by the participants, such as using temporal relations representation Jakkula & Cook (2011). To ensure a level of validity to these anomalies, the burden of defining what an anomaly is, should be left for the participants in the generation of the datasets to decide.

The issue of defining what an anomaly is can be relatively easy such as in the work done by Jain *et al.* (2006), where they used simple z-scores to detect anomalies in blood pressure and heart rate for the inhabitants. An anomalous blood pressure is medically defined. Thus, the use of simple statistical tests were sufficient.

Making a smart home aware of the user's context requires a level of intelligence. Current machine learning techniques fall short from reaching the full potential of anomaly detection and context-awareness. However, the human brain is more than capable of discerning context, and maybe machine learning techniques that simulate what happens in the human brain could help pushing the research towards intelligent anomaly detection. Recent advances in Neuroscience could assist in figuring out what makes us intelligent agents and also aid in developing more intelligent machine learning algorithms.

The use of HTM theory and the CLA to detect anomalies in a smart home setting has not been studied in the literature due to the lack of good datasets. This research project will try to fill some of the knowledge gaps in this area by testing the CLA on several datasets generated specifically for anomaly detection.

The IoT paradigm envisions the smart home to be filled with smart devices performing many functionalities. To study such environment, there has to be a test bed that can be used to simulate what will happen in that situation. Building a real world smart home with the characteristics that the IoT envisions is infeasible and costly for this research. Therefore, there is a need to build a virtual smart home environment that can be used to simulate typical daily activities of a smart homeowner. Moreover, having a virtual environment will give the researcher the flexibility to test different ideas and situations with minimum cost.

One of the identified gaps is the lack of a standard dataset that can be used as a benchmark for anomaly detection. Most of the real-world datasets available in the literature are not built with anomaly detection as their focus, and usually artificial anomalies are injected into the datasets after the real data have been collected.

Detecting anomalies is a challenging task. By definition, anomalies are rare and abnormal events. From this realisation and from the lack of real datasets of smart homes in the IoT era, the need to develop a testing and simulation environment became apparent. Thus, one of the goals of this project is to develop a simulation software that mimics the envisioned IoT smart homes and create test beds that will allow the researcher to evaluate the CLA performance in detecting anomalies.

Another identified issue related to the evaluation methodologies. The anomaly detection in the domain of smart homes studies usually use traditional evaluation methods such as recall and precision. These methods do not take into account the role of time in the evaluation process. Early anomaly detection is better than late detection and should be rewarded. Therefore, this research will investigate an evaluation method that takes the time aspect when scoring the performance of a proposed anomaly detection technique.

The research at hand will investigate the use and application of the HTM theory in order to detect abnormal behaviour generated by inhabitants of a smart home environment and will propose a novel anomaly detection model. The definition of an abnormal behaviour is a tricky subject and it is addressed in Section 6.2.2.2. Moreover, the anomaly detection is performed in an unsupervised fashion and only relies on the raw readings form the sensors and/or the smart devices.

2.9 Summary

In this Chapter, the IoT vision and paradigm and the major issues facing the realisation of the IoT vision along with the enabling technologies were presented. A review of the

middleware solutions available and how these middlewares model context and facilitate context-aware applications was presented and explored.

The challenges facing anomaly detection techniques were discussed along with the difficulty associated with defining what an anomaly is and how it is highly contextual and domain dependent.

A review of anomaly detection using different techniques was carried out and their advantages and disadvantages were discussed. The reviewed techniques cover classification, nearest neighbour, clustering, statistical, and spectral techniques. A more focused review was dedicated towards the unsupervised anomaly detection techniques because of how relevant these techniques are to detecting anomalies in the smart home domain and the scope of this research. The applications and domain of anomaly detection techniques were reviewed and grouped into the following domains: intrusion detection, fraud detection, health, text, image, video, and wireless sensor networks.

The smart home, as a prominent application of the IoT vision, was defined and reviewed. As well as the current applications and projects available with focus on anomaly detection techniques. A critical evaluation of the requirements to have an anomaly detection techniques in a smart home setting was presented.

The HTM theory, the CLA and NuPIC were reviewed and an initial experiment was conducted to quickly test and evaluate the potential of the CLA to detect anomalies in a smart home setting.

The Chapter concludes with the identified research gaps in the literature for anomaly detection in the smart home domain.

Chapter 3

Hierarchical Temporal Memory

3.1 Introduction

In this Chapter, Jeff Hawkins's theory of how the neocortex works will be presented (Hawkins & Ahmad, 2016). The theory is known as the HTM theory or sometimes referred to as the sequence memory theory. This Chapter starts with an introduction to the current scientific understanding of Neuroscience of how the brain biologically works and how the HTM theory is influenced by recent advancements in this field. The HTM theory focuses on the neocortex specifically because it is envisaged to be where our intelligence resides. The principals of the HTM theory will be explored and from these principals the CLA is built. The main components of CLA will be presented, including the Encoder, the Spatial Pooler (SP) and the Temporal Memory (TM). The CLA learns the spatial features of its inputs and predicts the transitions from one input to the other. The Chapter will be concluded by a review of several actual implementations of the HTM theory that are currently available for the research community.

3.2 HTM Theory

The neocortex is the outer wrinkly surface of the brain. It is around 2 to 4 millimetres thick and has billions of brain cells, also known as neurons. The neurons are vertically connected in a columnar structure, and they form a huge and complicated network of hierarchical regions. A famous study done by (Felleman & Van Essen, 1991) studied the layout of the regions associated with vision in the macaque monkey and as shown in Figure 3.1 the study identified many regions of neurons that are connected in a hierarchy. Each region receives information from the region below and passes it up to higher

regions. These regions are similar in structure and each region can be divided into six horizontal layers of neurons as shown in figure 3.2. These layers are named layer 1, 2/3, 4, 5 and 6. Layer 2 and 3 are usually grouped into one layer. The connections between the neurons go both ways, horizontal and vertical.

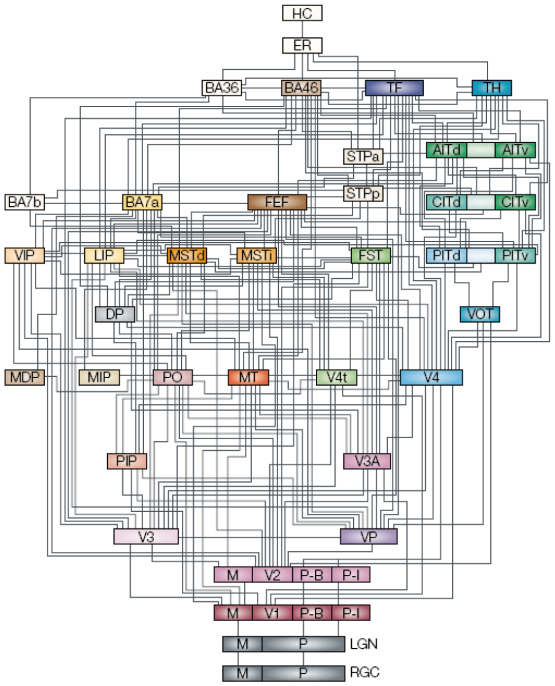


FIGURE 3.1: The neocortex regions of the visual system in the macaque monkey (Felleman & Van Essen, 1991).

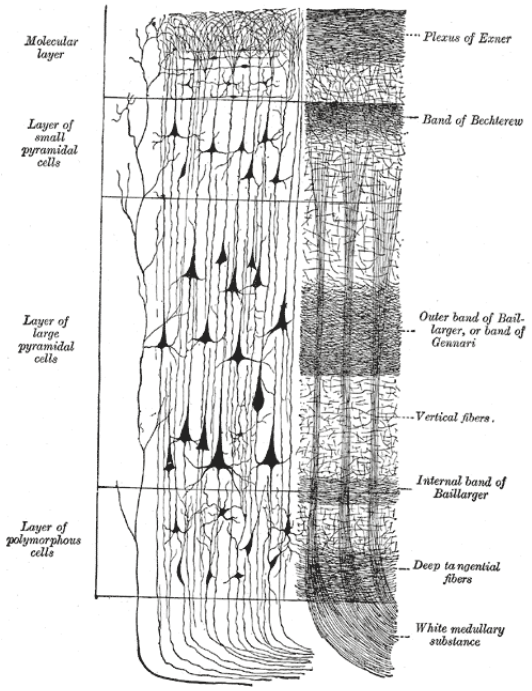


FIGURE 3.2: The layers of the neocortex from Gray's Anatomy (Williams *et al.*, 1980).

The neuron structure in general is shown in figure 3.3. The main components of a neuron are the cell body (the soma), the dendrites (the branches near the cell body), the axon and the synapse (the gap between the tip of the axon of the previous neuron and the dendrites of the next neuron). The synapse gap is changing constantly over time and it can grow and the gap becomes smaller, meaning the two cells have stronger connections. Or it can shrink and the gap widens and the connection is weaker. These changes are known in Neuroscience as *Neuroplasticity* and these changes are mainly responsible for learning. From this concept comes the idea of the weights of the cells in artificial neural networks (ANNs). For every neuron, there is a number of other neurons that could form connections. The synapses could be fully connected and could be fully disconnected.

There are many types of neurons in the neocortex but in this Chapter the focus will be on two main types, the *pyramidal* and the *interneuron* neurons. Pyramidal neurons excite and charge neurons, while interneurons do the opposite, they inhibit neurons. Both types perform similar tasks. When signals are coming to a neuron, they will charge the neuron and cause it to fire a signal (an action potential) that travels from the cell body through the axon until it reaches the tip of the axon. The difference between the two types comes from what they emit to the next neuron. Pyramidal neurons will act as *neurotransmitters* that cause the next neuron to charge up. While *interneurons* will emit another type of neurotransmitters that will inhibit the next neuron from firing signals. The dendrites on the next neuron receive this signal.

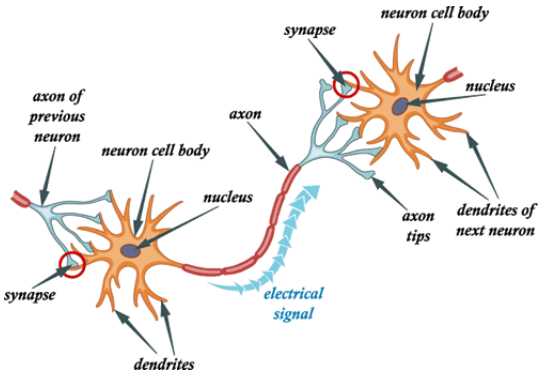


FIGURE 3.3: The neurons structure (Devineni, 2015).

The charging process is complex. When a neuron receives a signal, it will not always cause the neuron to fire unless some conditions are met. Also, the neurons do not hold the charge for a long time. If they do not receive enough charges in certain time interval, the neuron will not fire and the charge will fade away.

There are two types of inputs coming to every neuron. The input coming to the neuron through the *proximal* dendrites (the dendrites near the cell body) and the input coming

through the *distal* dendrites (the dendrites far away from the cell body). The proximal dendrites receive feed-forward signals coming from the senses and lower regions. Charging happens here in a linear fashion, up until it passes a threshold which will then cause the cell to fire a signal to all connected neurons.

On the other hand, the input coming through the distal dendrites comes laterally from other neurons, mostly from neurons in the nearby region. If enough signals are received to pass the threshold in a short time period, and also in a close segment of the dendrite, this will cause a dendritic signal that will *depolarise* the neuron.

As mentioned earlier, the synapse is the gap between a previous cell and the dendrites of the next cell. The synapses can be thought of as the connections between two neurons. Around 90% of the synapses are formed on the distal dendrites and the rest are formed on the proximal dendrites (Hawkins & Ahmad, 2016). The signals coming from the proximal dendrites affect the cell in a major way. On the other hand, signals coming from the distal dendrites seemed to have little effect on the cell activity (Major *et al.*, 2013). There have been studies that suggested an active role of the distal dendrites as processing units (Antic *et al.*, 2010; Major *et al.*, 2013). When several distal synapses located closely in time and space are activated, they can generate a local dendritic N-methyl-D-aspartate (NMDA) spike which will cause the cell to sustain depolarisation for longer time. This suggestion motivated more studies to investigate the role of the distal dendrites and whether they can function as an independent processing units that recognise patterns (Polsky *et al.*, 2004; Poirazi *et al.*, 2003).

Not all the synapses are connected all the time. Actually, the strength of these connections changes quite fast and ranges form fully connected to fully unconnected synapses. Furthermore, the synapses can only form up to certain distance which means that for a given neuron, there is a group of synapses that *can* form connections to it. These synapses are called *potential synapses* to the given neuron.

In Neuroscience, there is no prevailing and accepted theory of how the neocortex works. There are much evidence and large number of studies that focus on small aspects of the neocortex, but there is no overarching theory so far. Jeff Hawkins presented his HTM theory (also known as the memory-prediction theory) in his book 'on intelligence' (Hawkins & Blakeslee, 2004). The theory tries to provide a top-down approach of how the neocortex works. According to Hawkins, intelligence is fundamentally all about predictions. The memory-prediction theory of intelligence takes into consideration three factors that Hawkins believes are crucial for intelligence. The three factors are:

- Time,

- Feedback,
- The physical hierarchical structure of the brain.

Hawkins criticises most of the neural networks because of the over simplification of what really happens in the brain (Hawkins & Ahmad, 2016). Often one or more of the three factors are not/or poorly considered. For example, the cell body receives its input from the proximal dendrites near the cell body. These dendrites constitute about 10% of the number of all the dendrites in the neuron. The other 90% are distal dendrites. Despite the huge differences in numbers between the two types of dendrites, usually the neural networks model of a neuron does not take into account distal dendrites. They are ignored and the focus will be on the weights and strengths of the connections between neurons in the network.

The neocortex deals with many complicated problems, such as natural languages, vision, touch and hearing. Each functionality has regions in the neocortex associated with it. Yet, all of these functions seems to be performed via a common process in each region. There are many evidences from Neuroscience that suggest that there is a common algorithm performed by each region in the neocortex. This is what led to the development of the CLA which will be explained in the following Sections.

3.2.1 HTM Principles

The HTM theory has the following principles: hierarchy, regions, sparse distributed representations and time.

3.2.1.1 Hierarchy

The HTM system is essentially a memory system similar to the brain. There is no special part that stores data and another part that processes it. This memory system is organised in a hierarchy of regions. Each region is a collection of cells (neurons) arranged in columns. This memory system is different from the traditional flat memory system used in computers today. Of course, it is possible to simulate the hierarchy in software but this will add an overhead. The Von Neumann architecture (Von Neumann, 1993) of computers today could be improved to take into consideration a hierarchical memory system such as this.

3.2.1.2 Regions

As shown in figure 3.1, there are regions in the brain that are responsible for a given functionality. The same idea is presented here in the HTM. Each region is a collection of layered neurons arranged in columnar structure. The lower regions usually receive input from the senses. The output of the region is the input to the higher regions in the hierarchy.

3.2.1.3 Sparse Distributed Representations

As mentioned in the previous Section, the output of a region is the input to a higher region. But what is the nature of this input and output? In each region at a given time, a sparse formation of active neurons is happening. When looking at any region at any time, something similar to what is shown in figure 3.4 will be seen. Approximately, only 2% of neurons are in an active state at a given time. These formations or representations are actually one of the cornerstone ideas of the theory and are called the Sparse Distributed Representations or shortly, the SDRs.

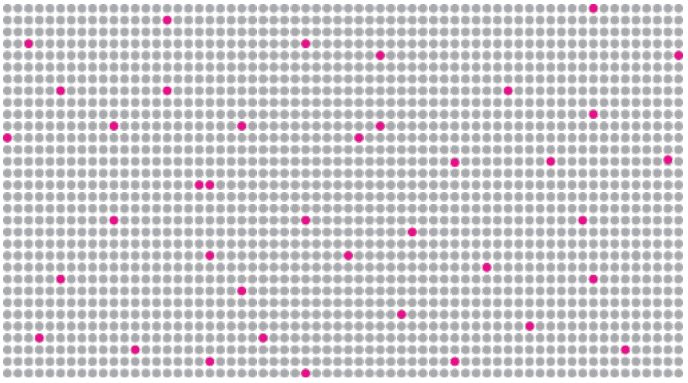


FIGURE 3.4: Sparse Distributed Representations (SDRs).

3.2.1.4 Time

The HTM system is actually a *sequence* of memories and time plays an important role here. Here is a thought experiment to explain the importance of time in this theory, imagine a blindfolded subject is asked to identify an object that is being placed on their palm. Assuming the object is an apple, that is placed on the subject's hand, and they are asked to identify it without moving their hand and feeling the object. It will be nearly impossible to identify the object as they need to feel and touch the object to be able to correctly identify it. These sensations are essentially sequences of SDRs travelling from the senses to the different regions in the neocortex. The same happens to hearing,

one cannot identify a song by only hearing a single note of a melody. Time is needed to construct the memory sequence of that sound. However, it is not that apparent in vision. The eyes are actually constantly moving in what is known as a *saccade* but this subtle and rapid movement of the eyes is not noticeable.

3.2.2 The Neurons in HTM Systems

The HTM neuron model is more complex than their usual counterparts in ANNs as shown in Figure 3.5. The ANN neurons usually compute the sum of all the weights of the connected synapse to that neuron. This concept, which is referred to as a 'point neuron' is the basic building block for most of the ANNs used in spiking neural networks (Maass, 1997) and deep learning (LeCun *et al.*, 2015). Figure 3.5 (A) shows a representation of a point neuron.

Figure 3.5 (B) shows a real neuron and the three sources of input it can receive. Namely, feed-forward input coming for lower regions (highlighted in green), context input coming laterally from neighbouring neurons and feedback input coming from higher regions. Figure 3.5 (C) shows an HTM neuron and how the three inputs affect the neuron. In green, the connections coming through the proximal dendrites (feed-forward) and in blue the connections coming through the distal dendrites (context) and the apical dendrites (feedback). The filled circles indicate that a signal was fired to the cell from another cell that is connected to the dendrite. When enough active dendrites are formed on the feed-forward input, the neuron (the grey triangle) will become active and fire an action potential. The feedback input coming to the neuron from its apical dendrites in higher regions biases the neuron to certain input by depolarising the cell (Spruston, 2008). The input values coming from the distal or the apical dendrites depolarise the neuron without causing it to fire an action potential (Rah *et al.*, 2013; Petreanu *et al.*, 2009; Yoshimura *et al.*, 2000). Hawkins & Ahmad (2016) claim this depolarisation is a mechanism for prediction that this cell is making. In other words, when enough signals are causing depolarisation, the cell is predicting it will be active in the next time step. It is a bias mechanism for the cells in reaction to the received input. Several studies showed that the activation of 8-20 synapses close in time and in space will cause an NMDA spike (Major *et al.*, 2013; Schiller & Schiller, 2001; Schiller *et al.*, 2000; Larkum *et al.*, 1999). Unlike the inputs coming from the proximal dendrites, these synapses combine their signals non-linearly till they exceed a threshold. The number of these firing synapses seems small to recognise large pattern reliably. But if the spareness aspect of these input values was added, then the recognition process will be robust. As suggested by Olshausen & Field (2004), sparse encoding seems to be a prevailing mechanism in sensory neurons. The HTM theory uses sparse distributed representation (SDRs) to model all the inputs.

The SDRs have many interesting mathematical properties that can be seen in details in Chapter 5.

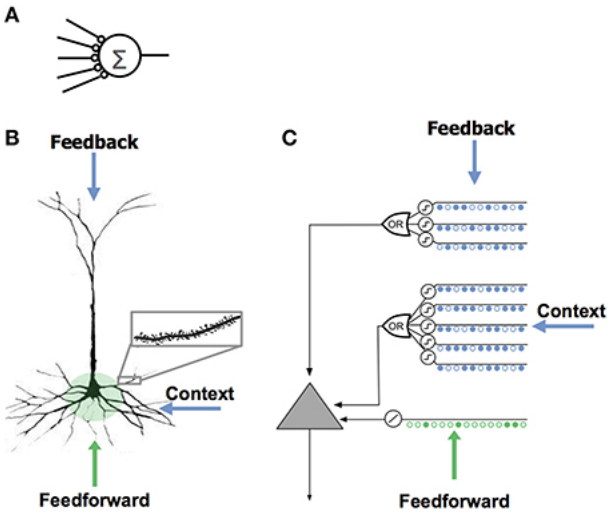


FIGURE 3.5: An HTM neuron and a real neuron (Hawkins & Ahmad, 2016).

3.2.2.1 HTM Neuron Inputs

Hawkins & Ahmad (2016) propose three zones to be the source of input to an HTM neuron. Namely, the proximal, basal and apical zones. As shown in Figure 3.5 (C), these zones correspond to feed-forward, context and feedback input respectively.

The proximal zone input has several hundred synapses that define the receptive field of the neuron (Spruston, 2008). The synapses in this zone have a large effect on the neuron to fire an action potential. The synapses in this zone learn to recognise feed-forward patterns coming through the cell receptive field.

The basal (or distal) synapses form lateral connections to nearby cells in region. They anticipate the neuron activation by recognising other neurons activities in proceeding time steps. Once a pattern is detected on the basal dendrites, they emit an NMDA spike which depolarises the neuron but not to the point of firing an action potential (Major *et al.*, 2013; Antic *et al.*, 2010). This mechanism gives an upper hand to the depolarised neuron to be active in the next time step if enough feed-forward input received. To Hawkins & Ahmad (2016), the basal synapses are responsible for learning the transitions of patterns in time.

When recognising some patterns, the dendrites formed on the apical zone of a neuron also generate NMDA spikes (Cichon & Gan, 2015). The NMDA spike coming from the apical dendrites affects the cell body indirectly. They cause a Ca^{2+} spike in the apical dendrite (Golding *et al.*, 1999; Larkum *et al.*, 1999). Just one of these Ca^{2+} spikes is

enough to depolarise the neuron but not enough to cause an action potential (Antic *et al.*, 2010). Although, this is an area under research (Larkum, 2013), to Hawkins & Ahmad (2016) this is another form of prediction similar to the one performed by the basal dendrites. This apical prediction provides a mechanism for higher regions to send feedback expectations to the lower regions and to bias them towards a certain activity. The CLA currently does not implement this feedback mechanism and it is under active research.

3.3 Cortical Learning Algorithm

The previously presented HTM neuron model is the theoretical HTM neuron. This section presents the algorithmic implementation of the HTM theory which is known as the CLA. It is worth noting that the concept of the neuron in the CLA is based on the previously explained HTM neuron model. In this section when the term "neuron" is used, we refer to the CLA neuron.

Figure 3.6 shows an overall view of how an HTM system works. The system receives streaming input data. The input data can be of any type, numerical, categorical or others. There must be an appropriate *encoder* for that input data (For more details on encoders see Chapter 5). The encoder then produces an SDR representing that input instance. The SDR is fed to the cubical structure shown in Figure 3.6 which is a collection of cells (neurons). This cubical structure is the HTM model of layer 2/3 of the neocortex. The input to this cubical structure is received by the *Spatial Pooler (SP)* which operates on the columns in that structure. Then, the *Temporal Memory (TM)* will operate on the columns by activating and deactivating the cells within the columns. These columns are referred to as the 'mini-columns' in the HTM literature. The CLA is the algorithm that describes how the SP and the TM work. Currently, the CLA implements some parts of the HTM theory and some aspects are still under active research.

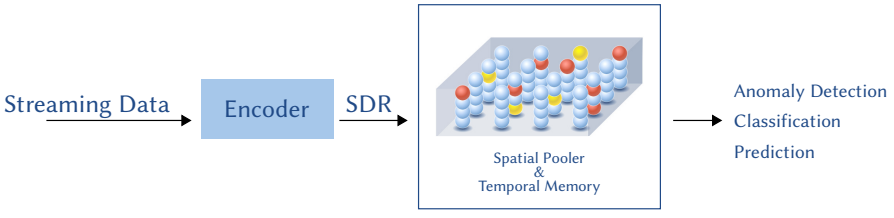


FIGURE 3.6: An HTM system workflow.

In the CLA, the SDRs are represented as a sequence of 1's and 0's. At any given time, an SDR will have around 2% of its bits active. This is an important property of the

SDRs and of the whole CLA and leads to many interesting results (More details are in Chapter 5). Another important difference between the SDRs and, for example, the usual representations (encodings) in computers today, is that every bit is not significant. For instance, when taking the ASCII encoding and changing one single bit, this will create a new and totally different meaning of the code. This is not the case in the SDRs. If you change one bit in an SDR, it will not have any effect on the representation. This makes the SDRs robust to noisy data streams.

The SDRs are first created by an encoder that receives the input and converts it to a sparse array of active and inactive bits. Then, the SDR is fed to the SP which performs several operations based on the SDR configuration and consequently the output of the SP is fed in to the TM. The output of the TM is a collection of inactive, active and predictive cells that represent the knowledge that the HTM system learned from the world by receiving this sequence of SDRs.

Every neuron in the CLA can be in one of the following three states, inactive, active and predictive. The inactive state indicates that the cell did not yet receive enough signals either from its proximal or distal dendrites. The active state indicates that the cell is active due to signals coming from the proximal dendrites. The predictive state is caused by receiving enough signals on the distal dendrites as discussed earlier.

In the CLA, the synapses have binary weights, either connected or not. The strength of the connections of the synapses is a ratio ranging from 0 to 100%. In several places in the CLA, there are thresholds or *permanences* of each synapse. If the strengths of the connections passes these thresholds, then the synapse will be connected.

3.3.1 The Algorithm

The CLA algorithm input is an SDR generated by an encoder that corresponds to a certain record in a dataset. The output of the CLA is a collection of inactive, active and predictive cells.

The cubical structure shown in Figure 3.6 is the CLA main data structure. It is composed of mini-columns and single cells (neurons). The SP operates based on mini-columns and the TM operates on individual cells. This arrangement allows the HTM system to simultaneously represent the feed-forward input and the context input. Figure 3.7 shows the same cubical structure but in a 2 dimensional configuration to make it easier to see the individual states of the cells.

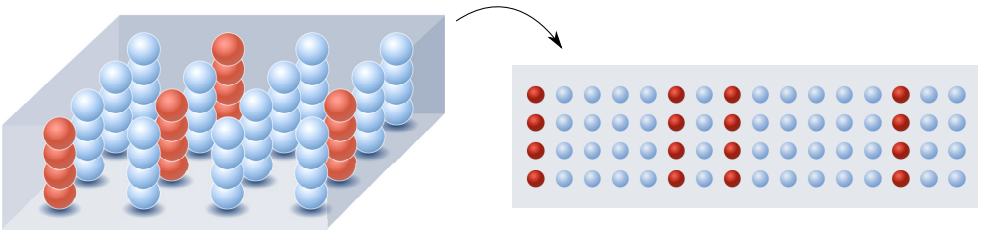


FIGURE 3.7: Flattened neurons in an HTM system.

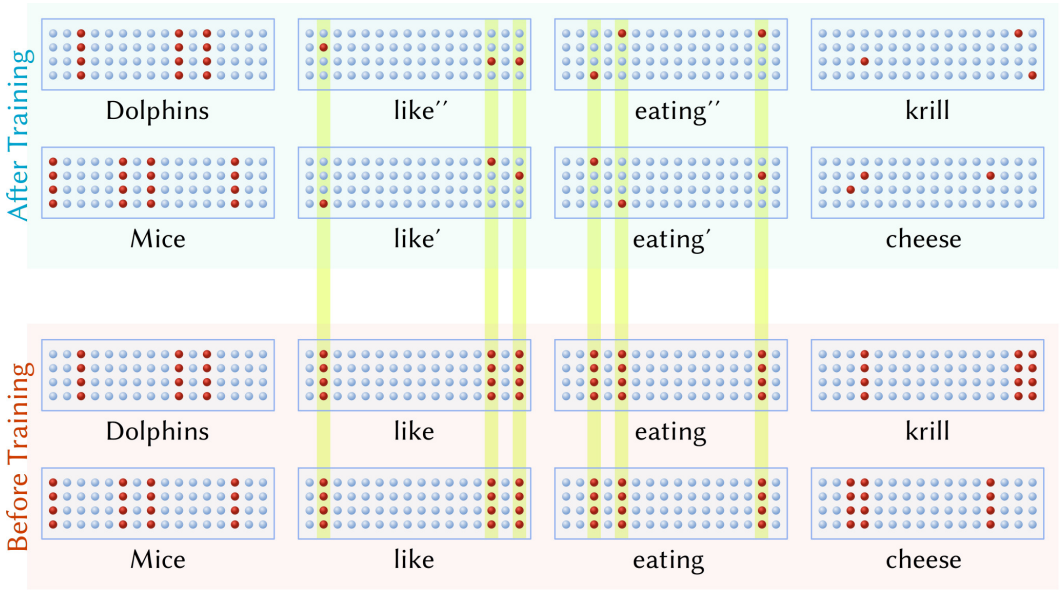


FIGURE 3.8: Two sequences and how they are represented in the CLA over time.

To explore how the CLA works, let us consider two sequences, $Mice \rightarrow like \rightarrow eating \rightarrow cheese$ and $Dolphins \rightarrow like \rightarrow eating \rightarrow krill$. Each word will be fed to the HTM system one at a time. Figure 3.8 shows the two sequences and each word representation. Notice that the sub-sequence $like, eating$ is presented in both sequences. The figure shows two phases, before and after training for both sequences. Every word has an encoding of different mini-columns, similar to barcodes. Now assuming the system has learned the two sequences, and it is now being tested on new data. When the system first sees the words $Mice$ or $Dolphins$, it will anticipate that the word $like$ is likely to follow. The issue here is that there are two $like$'s. How can the system distinguish between the first $like'$ from the second $like''$ which is in a different context both sequences. The algorithm here uses single cells in the same mini-columns that represent the word $like$ to encode for the different contexts that the word $like$ appears in. This gives the CLA a huge capacity to remember long sequences in different contexts while retaining the mini-columnar representation of each word.

Notice that after the system has been trained, the first word for each sequence always activates all the cells in the mini-columns. This is the HTM system's way of representing

an unknown sequence and this representation is referred to as *bursting columns* in the HTM literature. Since the system did not see the words $Dolphins$ and $Mice$ in any prior context it will *bursts* all the mini-columns.

Now let us consider the following case. Suppose that the HTM system already learned the two previous sequences. Figure 3.9 shows how the HTM system performs predictions for the subsequent words. If the system first sees the word $Mice$, it will predict the following word to be $like'$. Notice the second panel in Figure 3.9 that the predicted cells (yellow cells) are the ones corresponding to the word $like'$ in the sequence $Mice \rightarrow like \rightarrow eating \rightarrow cheese$. If the following word in the sequence is indeed $like'$ as shown in the third panel, the system will predict the following word to be $eating'$ and so on. Whenever the system correctly predict a word, its belief and understanding of the world will be stronger and the synapses will become more connected. On the other hand, whenever a prediction is wrong, the synapses will be weakened. The fourth panel in the same figure shows the distal segment (in green) for one of the cells that learned this sequence. These segments will be formed and destroyed throughout the life of an HTM system, and their connections will be strengthened and weakened depending on whether the system's prediction are correct or not.

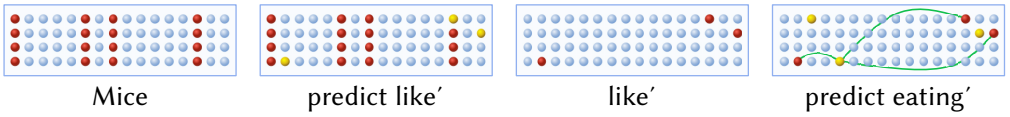


FIGURE 3.9: Predicting the next word in a learned sequence.

Another interesting scenario can happen when feeding the system the sequence $like \rightarrow eating$ as shown in Figure 3.10. The system will not recognise the word $like$ coming as the first word in the sequence. It knows the word $like$ but this sequence is novel to the system. This will make the system burst all the mini-columns that are corresponding to the word $like$. More interestingly, the system will predict both words $eating'$ and $eating''$ since it does not know if $eating$ will be the one coming for the $Mice$ sequence or the $Dolphins$ sequence. As shown in the second panel of the same figure, both words cells will be in a predictive state. When indeed the next word is $eating$, the system still does not know which word will come after that. Is it $cheese$ or $krill$? Therefore, the system will predict both words. Notice that some cells can be in two states at the same time. They can be active and predictive as shown in one of the cells in the fourth panel.

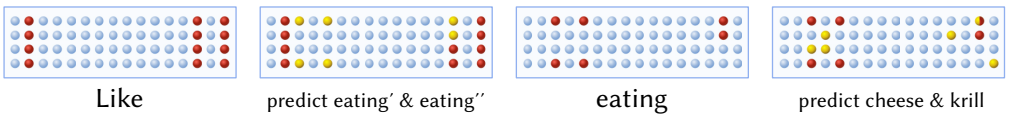


FIGURE 3.10: Predicting multiple ambiguous words in a learned sequence.

Here is another look at the same process but using the cubical structure presented in Figure 3.6. In Figure 3.11 (a), we see one layer or a slice of that cubical structure (not to be confused with the six layers in the neocortex) of a region and each cube represents a neuron. The layer receives input from lower regions and the colour intensity corresponds to the cell charge, the darker the colour is, the more likely the cell will fire. In (b), a cell is passing the threshold of firing. When this happens, the cell will *inhibit* the neighbouring cells and reset their charges. The area of the inhibition is configurable. It can cover part of the available space as shown in the figure or it can cover the whole space. When the inhibition covers all the available space it is referred to as *global inhibition* within the HTM literature. Figure 3.11 (c), shows the result of the whole process, a sparse and distributed representation.

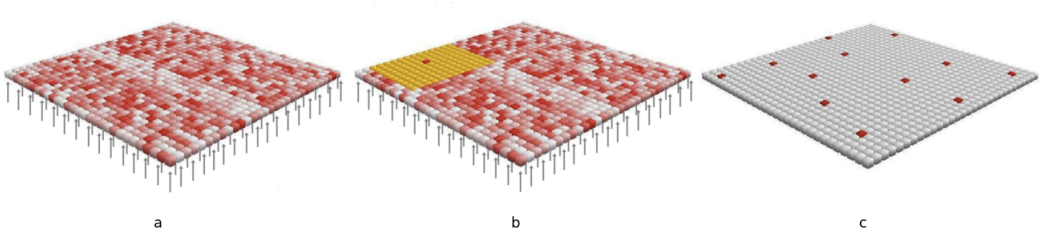


FIGURE 3.11: The CLA performing inhibition (Jeff Hawkins, 2014).

Figure 3.12 shows what happens in a single layer over time. In (a), an SDR at a given time step followed by another SDR at the next time step (b). When a cell becomes active in the next time step, it strengthens and/or makes new connections with the neighbouring cells that were active in the past time step as shown in (c). These connections are the 'segments' for this current input. What this means is, in the future, when the highlighted cell in (c) sees that the cells that it is connected to became active, this is like a signal for the highlighted cell to anticipate and prepare itself to be active in the next time step. This is where the predictions happen in the SDRs. So ideally at any given time, looking at a layer, the cells in this layer should have around 2% of the cells active, and other cells in a predictive state for the next input. If the predictions are correct, the strength of the connections will be increased and vice versa. This is shown in (b), the red cells are the active cells, and the yellow cells are the predictive cells. Note that the segments are formed on the same layer and on any layer in the region.

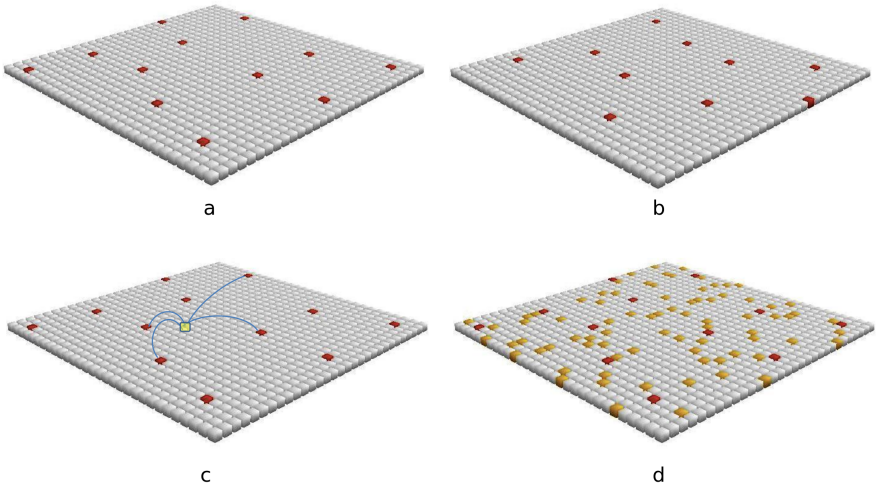


FIGURE 3.12: The CLA changes over time (Jeff Hawkins, 2014).

If only one layer is used in the HTM system, it will only be able to *remember* one step back in history. It does not have the capacity to remember longer sequences. To solve this issue, the HTM systems usually have multiple layers, similar to the columnar structure of the real neurons as shown in figure 3.13.

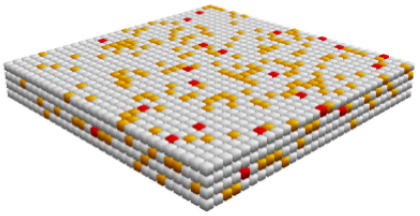


FIGURE 3.13: Higher order memory(Jeff Hawkins, 2014).

Overall, after the encoder converts the input into SDRs, the CLA process can be summarised by three steps:

- Learn the spatial features of the input SDRs and convert it to a sparse collection of active mini-columns (which are considered as SDRs if the mini-columns are viewed as active cells),
- Learn the transitions of these SDRs over time and activate cells that represent these transitions,
- Predict the next coming SDR by learning from the past transitions and put cells in a predictive state.

To achieve these steps, the CLA uses two major components in its algorithm which are the SP and the TM. Both of these components have similar concepts. Learning, in essence, is performed by creating and/or destroying synapses (or connections) from a potential collection of synapses. A synapse is considered connected when its permanence exceeds a certain threshold. The value of the permanence is updated in a Hebbian-like rule (Hawkins & Ahmad, 2016). Therefore, the synapses have binary weights unlike many ANNs where the weights are scalar values.

The synapses in HTM systems form connections to dendrite segments. Currently, the CLA implements two dendrite zones, the proximal and the distal/basal dendritic zones. The proximal segment establishes connections with a subset (receptive field) of the feed-forward input. When there are enough active synapses formed on this subset of the input, the cell will be activated. The number of active synapses are linearly summed up. The distal segment forms lateral connections with nearby cells in the same region. When there are enough active synapses on the distal segments, the cell will be put in a predictive state. For a single cell, there are many distal segments available. Therefore, the logical operation `OR` will be performed on the active synapses on all distal segments for this cell.

In the following two subsections, the SP and the TM will be discussed in more details.

3.3.2 The Spatial Pooler

The main function of the SP is to take the input from the encoder and convert it to a sparse distributed representation. The name of the SP comes from the fact that it is trying to pool or group inputs that are spatially similar into one representation. The input from the encoder can be sparse or not. The SP is responsible for ensuring the sparsity of this input. Figure 3.14 shows the SP activating a subset of the mini-columns. The highlighted active mini-column is connected to a subset of the input space (the output of the encoder). The radius of this potential connection (or the receptive field) can be configured, but generally it is not covering the whole input space. Therefore, a mini-column will have a natural centre with the input space underneath it.

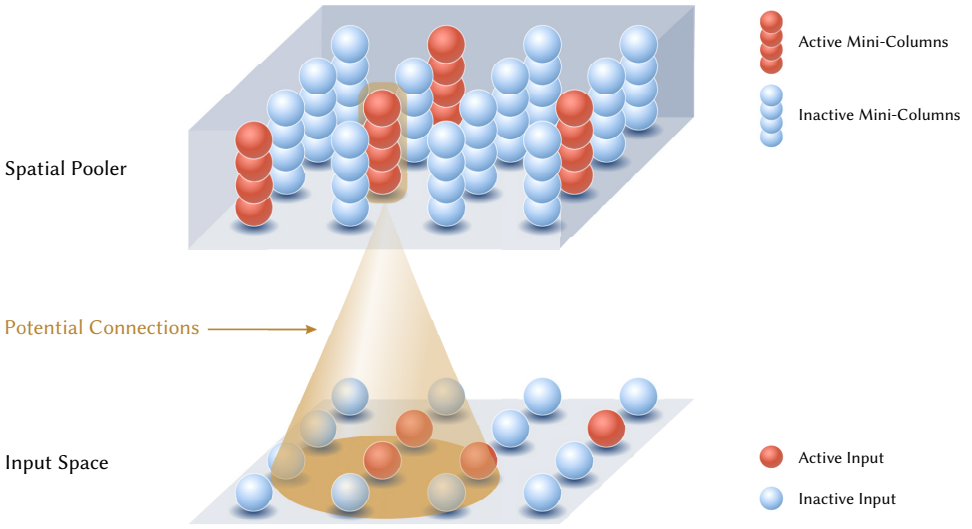


FIGURE 3.14: A mini-column in the SP and its receptive field.

The SP uses a fixed number of mini-columns which need to be setup before training the HTM system. Therefore, the SP must ensure the best utilisation of this fixed amount of resources. One way to achieve this, is to make sure that a large number of the mini-columns are contributing in the learning process. In the SP, this is achieved by implementing a 'boosting' mechanism that makes the losing mini-column more aggressive and compete to represent the coming inputs. Determining the level of activity for a certain mini-column is done by measuring the activity of a given mini-column and its neighbouring mini-column activities. Over time, the mini-columns will be specialised and more sensitive to certain SDRs.

The inhibition process will limit the number of the actual winning mini-column to a fixed sparsity. As shown in Figure 3.11, the number of the winning mini-columns will stay constant throughout the life of the HTM system. This fixed number is one of the configurable parameters that an HTM user can change and experiment with.

The SP can form stable representation of its input. A mini-column need to have several active synapses active to become active itself. Controlling the number of the needed active synapses is a way to avoid learning noisy data. It could be the case that one mini-column has too many true connections, to counter this issue, the SP will decrease the permanence value for the synapses which are not contributing to a winning mini-column.

The SP has these balancing measures that ensure the contributions of all available mini-columns. These aforementioned measures make the HTM system flexible and adaptive to the input it is receiving. This is an important quality to have for on-line machine

learning systems. This flexibility and plasticity has its inspiration from the brain plasticity and how regions of the brain can adapt to perform other functions when damage happens.

Here is a high level description of the steps the SP algorithm performs:

1. Receive a one dimensional array of bits generated by an encoder or by a lower region in the hierarchy. For the SP, this input is referred to as the *input space*.
2. Allocate a fixed number of mini-columns. Each mini-column has a potential connection to a subset of the input space. Each potential connection has a permanence value and this value will determine if the connection is active or not. That is, if the permanence value exceeds the connection threshold.
3. For a given SDR, calculate how many synapses are connected to an active bit in the input space. This is referred to as the *overlap score*.
4. Multiply the number of the active synapses by a boosting factor. As discussed before, this boosting factor is determined by how active a mini-column is. The less active a mini-column is compared to its neighbouring mini-columns, the higher the boosting factor.
5. After boosting, the mini-column with the highest number of active synapses will become active. The number of these winning mini-columns can be configured by the HTM system user through setting a threshold. This will effectively generate a result similar to what was shown in Figure 3.11. In other words, the winning mini-columns will inhibit their neighbouring mini-columns form winning.
6. Update several permanence values. The permanence value for the synapses that are connected to active bits in the input space are increased. The synapses that are connected (or potentially connected) to inactive bits in the input space will have their permanence value decreased. The increase and decrease in the permanence values will cause synapses to connect and disconnect to segments of the input space.

For more details about the actual algorithm code and the data structures used, see Appendix A.

3.3.3 The Temporal Memory

The TM main aim is to learn the transitions of the SDRs generated by the SP and predict future sequences. When a neuron or a cell becomes active, it forms lateral connections

with the cells that were active in the prior time step. In the future when these cells will be active, the cell that formed the lateral connections will be in predictive state. This process is the essence of learning the transitions in the TM. Figure 3.15 shows two formed lateral segment connections. Similar to what was done in the SP in terms of increasing and decreasing the permanence of these connections, the TM will increase the permanence of correctly predicted sequences and punish the wrong prediction by decreasing their permanence values. Therefore, the whole HTM system is a memory system that learns about the world and encodes this knowledge in the cells states. There is no storage unit that records the information the HTM learned about the world. Since this knowledge is distributed across many cells, the HTM system is robust to noise and there is no issue when a cell or several cells fail.

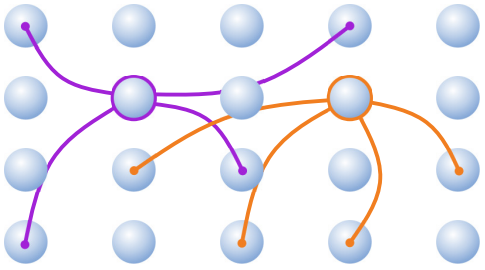


FIGURE 3.15: Two formed segments in the TM.

In Hawkins *et al.* (2016), there is an example that shows the importance of having sparse inputs and how the TM exploits the SDRs properties to reduce the memory footprint. The examples assume an HTM system with 10,000 cells out of which 200 are active at any given time. One might assume that for the TM to recognise an input, it needs to match all 200 active cells with what it knows. This is not the case, in fact, as little as 20 active bits are needed to recognise the input with very low probability of false positives. Therefore, the TM only needs a subset of these active bits, a process known in HTM literature as 'sub-sampling'. More on the SDRs properties are presented in Chapter 5.

Here is a high level description of the steps the TM algorithm does:

1. Receive the active mini-columns generated by the SP. For every mini-column, check if there are any cells in a predictive state and activate them. If no predictive cells are present in a mini-column, activate all the cells in this mini-columns.

2. For all the cells in the region, go through their dendrite segments and count the connected synapses of active cells. If their number passes a threshold, make this segment as an active segment. All the cells that have active dendrite segments are put into a predictive state. All the cells that have no active dendrites and are not active due to feed-forward input stay inactive.

3. Update the permanence values for any dendrite segment that became active. For all synapses in this dendrite segment, increase the permanence of the synapses connected to active cells and decrease the permanence of synapses connected (or potentially connected) to inactive cells. The list of permanence updates are put on hold and marked temporary until the next step is performed.

4. When a cell become active because of feed-forward input, all the potential synapses in this cell are searched and any temporary marks are removed. Therefore, the permanence updates are only applied when the prediction is correctly realised.

5. When a cell switches from being active to inactive, discard all the permanence updates for this cell because the prediction is wrong.

For more details about the actual algorithm code and the data structures used, see Appendix A.

3.4 HTM Implementations

The HTM theory inspired many projects and has many implementations currently available in the literature. In this Section, these projects will be presented and the results of some of these implementations performing anomaly detection on a dataset are reported.

The reference and state-of-the-art implementation for the HTM theory is Numenta Platform for Intelligent Computing (NuPIC). This implementation has the most active development and commercial products have been built on top of the code base. Here is a list of some products that are based on NuPIC:

- **HTM for stocks**: Detects anomalies in stock market.
- **Grok**: Detects anomalies in servers and applications.
- **Rogue behaviour detection**: Detects abnormal behaviour of individuals in a company such as accessing an unauthorised file or abnormal downloading activities.
- **Geo-spacial tracking**: Detects anomalies in geo-spacial data.

NuPIC is written in Python and C++ but there are projects developed by the community for other programming languages, such as htm.java[1] which is written in Java. Clortex is an implementation of HTM written in Clojure[2].

[1]https://github.com/numenta/htm.java
[2]https://github.com/htm-community/clortex

Comportex[3] is another implementation of the HTM theory written in Clojure. Comportex is not a re-write in another language implementation. It is an experimental implementation different from NuPIC.

Table 3.1 shows a comparison between NuPIC, htm.java and Comportex at anomaly detection using Numenta Anomaly Benchemark (NAB) framework (see Section 6.2.3.1 for more details). NAB has several real-world datasets with annotated anomalies and synthetic datasets as well (Lavin & Ahmad, 2015).

TABLE 3.1: HTM implementations anomaly detection score using NAB.

Implementation	**Standard Profile**	**Reward Low FP**	**Reward Low FN**
NuPIC	70.1%	63.1%	74.3%
HTM Java[4]	65.5%	53.2%	70.4%
Comportex[5]	64.6%	58.8%	69.6%

At the time of writing, NuPIC is considered the state-of-the-art of the HTM implementations. It is under an active development by researchers and members of the community.

3.5 Summary

In this Chapter, the current Neuroscience understanding of how parts of the neocortex works was presented. The HTM theory based on these evidences builds an overarching theory that proposes the existence of a common algorithm performed across all regions of the brain, as opposed to having an algorithm specific for each region of the brain. The CLA is proposed as a common algorithm that describes how the brain learns from its inputs. The HTM theory proposes a different and more realistic neuron model compared to the point neurons that are usually used in ANNs. The point neurons only model the feed-forward input as varying weights guided by the fact that feed-forward input coming from proximal dendrites has the largest effect on real neurons. However, the real neurons only have 10% of their dendrites located at the proximal zone. The other 90% of the dendritic connections are formed laterally to neighbouring neuron across the region. The HTM theory proposes that these 90% distal dendrites are playing an important role in performing predictions.

The CLA is under active development and research and currently implements only parts of the HTM theory principals. The SP and the TM are the two main components of this

[3]https://github.com/htm-community/comportex

algorithm. The SP learns how to group and pool similar inputs and learn their spatial features. While the TM is responsible of learning the changes or the transitions between these inputs.

There are several actual code implementations available for the CLA and the HTM theory in the literature. A review of these projects was presented in the context of anomaly detection. These projects were tested on NAB datasets and their scores were presented. As shown earlier, NuPIC is considered the state-of-the-art implementation of the HTM theory, hence it will be used and adopted for this study.

Chapter 4

OpenSHS

4.1 Introduction

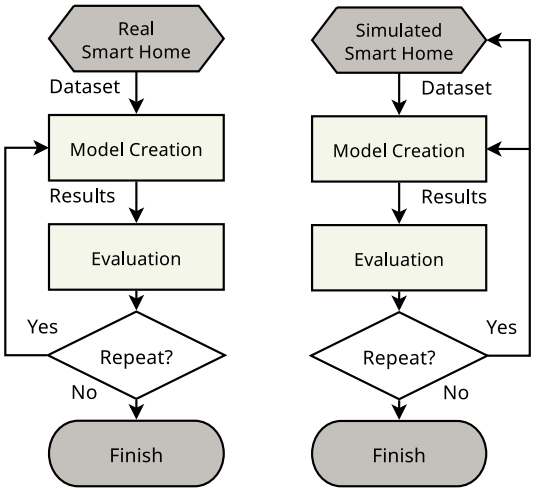
With the recent rise of the IoT, analysing data captured from smart homes is gaining more research interest. Moreover, developing intelligent Machine Learning techniques that are able to provide services to the smart home inhabitants are becoming popular research areas.

Intelligent services, such as classification and recognition of Activities of Daily Living (ADL) and anomaly detection in elderly daily behaviour, require the existence of good datasets that enable testing and validation of the results (Buchmayr *et al.*, 2011; Rodner & Litz, 2013; Youngblood *et al.*, 2005; Helal *et al.*, 2011). The medical field also recognised the importance of analysing ADLs and how these techniques are effective at detecting medical conditions for the patients (Tapia *et al.*, 2004). These research projects require either real or synthetic datasets that are representative of the scenarios captured from a smart home. However, the cost to build real smart homes and the collection of datasets for such scenarios is expensive and sometimes infeasible to many projects (Synnott *et al.*, 2015; Helal *et al.*, 2011; Mendez-Vazquez *et al.*, 2009; Lei *et al.*, 2010; Armac & Retkowitz, 2007). Moreover, several issues face the researchers before actually building the smart home such as finding the optimal placement of the sensors (Helal *et al.*, 2010), lack of flexibility (Armac & Retkowitz, 2007; Fu *et al.*, 2011), finding appropriate participants (Helal *et al.*, 2011; Mendez-Vazquez *et al.*, 2009), and privacy and ethical issues (Poland *et al.*, 2009).

Even though there exist real smart home datasets (Cook *et al.*, 2003b; Alemdar *et al.*, 2013; Munguia Tapia, 2003), sometimes they do not meet the needs of the conducted research project. Such as, the need to add more sensors or to control the type of the

generated scenarios. Very few of such datasets record the readings of the sensors in real-time and provide a detailed time-stamped field like the ARAS dataset (Alemdar *et al.*, 2013). Moreover, preparing a real dataset could be a laborious task, and if not done with care, it could lead to producing erroneous output.

When building real smart home testbeds, several challenges are facing the preparation of real datasets. One challenge is having a robust and continuous capturing mechanism for the sensors' data. Another challenge is following an appropriate annotation method for the inhabitants' activities.



(A) A real testbed. (B) A simulated testbed.

FIGURE 4.1: The workflow of real and simulated smart homes testbeds.

The existence of a dataset simulation tool overcomes the drawbacks/challenges of generating real datasets. Such tools facilitate fast dataset generation and offer robust methods to capture the sensors' data. Additionally, they can offer solutions such as the ability to pause and fast-forward the simulation to enable more accurate activity annotation. When developing Machine Learning models, targeting specific functionalities, researchers rely on the existence of good representative datasets. A common practice in Machine Learning is to divide the dataset into two parts, training and testing. The model creation starts by initialising its parameters and training on a portion of the dataset. Then, the model will be tested on another part of the same dataset and its results will be evaluated. The results of the evaluation could reveal the need to redesign the smart home by adding or removing smart devices or changing the scenarios generated, etc. In the case of a real smart home, if the results revealed the need to change something, this is usually a costly and infeasible choice to make. Therefore, the researcher could only be able to tweak the model parameters as shown in Figure 4.1a. On the other hand,

with a simulated smart home, this can be easily done, and the researcher can go back and modify the smart home design as shown in Figure 4.1b.

The approaches for the smart home simulation tools can be divided to model-based and interactive approaches. The model-based approaches use statistical models to generate datasets while the interactive approaches relies on real-time capturing of fine-grained activities using an avatar controlled by a human/simulated participant. Each approach has its advantages and disadvantages.

From what was mentioned earlier, it is apparent that the virtual simulation tool should offer far greater flexibility and lower cost than conducting an actual and physical smart home simulation (Synnott *et al.*, 2015). The new recent advances in computer graphics, such as Virtual Reality (VR) technologies can provide immersive and semi-realistic experiences that could come close to the real experience. The simulation tool should also be open and readily available to both, the researchers and the test subjects.

Although there are some research efforts available in the literature for smart home simulation tools, they suffer from some limitations. The majority of these tools are not available in the public domain as an open-source project, or limited to a particular platform. Also, most of the publicly available simulation tools lack the flexibility to add and customise new sensors or devices.

When generating datasets, the model-based approaches are capable of generating bigger datasets but the granularity of captured interactions are not as fine as the interactive approaches. However, the interactive approaches usually take longer time to produce the datasets as they capture the interactions in real-time.

This research proposes a novel smart home simulation tool called OpenSHS which is a new hybrid, open-source, cross-platform 3D smart home simulator for dataset generation. Its significant contribution is that OpenSHS offers an opportunity for researchers in the field of IoT and Machine Learning to produce and share their smart home datasets as well as testing, comparing and evaluating their models objectively. Following a hybrid approach, OpenSHS combines advantages from both interactive and model-based approaches. This method reduces the time and efforts required to generate simulated smart home datasets. OpenSHS includes an extensible library of smart devices that facilitates the simulation of current and future smart home environments. A replication algorithm for extending and expanding a dataset was designed. A small sample dataset produced, by OpenSHS, can be extended without affecting the logical order of the events. The replication provides a solution for generating large representative smart

home datasets. Moreover, OpenSHS offers a feature to shortening and extending the duration of the generated activities. In this Chapter, the architecture and implementation of OpenSHS is presented

The rest of this Chapter is structured as follows: the following Section reviews existing real smart home testbeds and simulation tools, this Section is concluded by analysing existing smart home simulation tools and comparing them with our proposed tool, OpenSHS. Section 4.3 presents the architecture of OpenSHS and its implementation. Section 4.4 presents two usability studies for using OpenSHS by researchers and participants. Followed by the conclusion of this Chapter.

4.2 Related Work

The literature is rich with efforts that focus on generating datasets for smart home applications. These efforts can be classified into two main categories, datasets generated either from real smart homes testbeds or using smart home simulation tools.

4.2.1 Real Smart Home Testbeds

One of the recent projects for building real smart homes for research purposes was the work carried out by the Centre for Advanced Studies in Adaptive Systems (CASAS) (Cook *et al.*, 2013) where they created a toolkit called 'smart home in a box' which is easily installed in a home to make it able to provide smart services. The components of the toolkit are small and can fit in a single box. The toolkit has been installed in 32 homes to capture the participants' interactions. The datasets are publicly available online (CASAS, 2009).

The TigerPlace (Skubic *et al.*, 2009) project is an effort to tackle the growing ageing population. Using passive sensor networks implemented in 17 flats within an elder care establishment. The sensors include motion sensors, proximity sensors, pressure sensors and other types. The data collection took more than two years for some testbeds.

SmartLab (Nugent *et al.*, 2009) is a smart laboratory devised to conduct experiments in smart living environments to assess the development of independent living technologies. The laboratory has many types of sensors such as pressure, passive infrared (PIR), and contact sensors. The participants' interactions with SmartLab are captured in an XML-based schema called homeML (McDonald *et al.*, 2013).

The Ubiquitous Home (Yamazaki, 2007) is a smart home that was built to study context-aware services by providing cameras, microphones, pressure sensors, accelerometers, and other sensor technologies. The home consists of several rooms equipped with different sensors. To provide contextual services to each resident, the Ubiquitous home recognises the resident by providing Radio-Frequency Identification (RFID) tags and by utilising the installed cameras.

PlaceLab (Intille *et al.*, 2006) is a 1000 sq.ft. smart flats that has several rooms. The flats has many sensors distributed throughout each room, such as electrical current sensors, humidity sensors, light sensors, water flow sensors, etc. Volunteering participant can live in PlaceLab to generate a dataset of their interaction and behaviour. The project produced several datasets for different scenarios (PlaceLab, 2005).

HomeLab (De Ruyter *et al.*, 2005) is a smart home equipped with 34 cameras distributed around several rooms. The project has an observation room that allows the researcher to observe and monitor the conducted experiments. HomeLab aims to provide datasets to study human behaviour in smart environments and investigate technology acceptance and usability.

The GatorTech smart home (Helal *et al.*, 2005) is a programmable and customisable smart home that focuses on studying the ability of pervasive computing systems to evolve and adapt to future advances in sensors technology.

4.2.2 Smart Home Simulation Tools

Smart home simulation tools can be categorised into two main approaches, according to Synnott *et al.* (2015), model-based and interactive approaches.

4.2.2.1 Model-Based Approach

This approach uses pre-defined models of activities to generate synthetic data. These models specify the order of events, the probability of their occurrence, and the duration of each activity. This approach facilitates the generation of large datasets in a short period. However, the downside of this approach is that it cannot capture intricate interactions or unexpected accidents that are common in real homes. An example of such approach is the work done by Mendez-Vazquez *et al.* (2009).

PerSim 3D (Lee *et al.*, 2015) is a tool to simulate and model user activities in smart spaces. The aim of this tool is to generate realistic datasets for complex scenarios of the inhabitant's activities. The tool provides a Graphical User Interface (GUI) for visualising

the activities in 3D. The researcher can define contexts and set ranges of acceptable values for the sensors in the smart home. However, the tool is not available freely in the public domain.

SIMACT (Bouchard *et al.*, 2010) is a 3D smart home simulator designed for activity recognition. SIMACT has many pre-recorded scenarios that were captured from clinical experiments, which can be used to generate datasets for the recognition of ADLs. SIMACT is a 3D open-source and cross-platform project developed with Java and uses Java Monkey Engine (JME) (JME, 2003) as its 3D engine.

DiaSim (Jouve *et al.*, 2009) is a simulator developed using Java for pervasive computing systems that can deal with heterogeneous smart home devices. It has a scenario editor that allows the researcher to build the virtual environment to simulate a certain scenario.

The Context-Aware Simulation System (CASS) (Park *et al.*, 2007) is another tool that aims at generating context information and test context-awareness applications in a virtual smart home. CASS allows the researcher to set rules for different contexts. A rule can be, for example, turn the air conditioner on if a room reaches a specific temperature. The tool can detect conflicts between the rules of the pre-defined contextual scenarios and determine the best positioning of the sensors. CASS provides a 2D visualisation GUI for the virtual smart home.

The Context-Awareness Simulation Toolkit (CAST) (Kim *et al.*, 2006) is a simulation tool designed to test context-awareness applications and provides visualisations of different contexts. The tool generates context information from the users in a virtual smart home. CAST was developed with the proprietary technology Adobe Flash and is not available in the public domain.

4.2.2.2 Interactive Approach

Contrary to the previous approach, the interactive approach can capture more interesting interactions and fine details. This approach relies on having an avatar that can be controlled by a researcher, human participant or simulated participant. The avatar moves and interacts with the virtual environment which has virtual sensors and/or actuators. The interactions could be done passively or actively. One example of passive interactions is having a virtual pressure sensor installed on the floor and when the avatar walks on it, the sensor should detect this and emit a signal. Active interactions involve actions such as opening a door or turning the light on or off. The disadvantage of this approach, however, is that it is a time-consuming approach to generate sufficient datasets as all interactions must be captured in real-time.

Park *et al.* (2015) presented a virtual space simulator that can generate inhabitants data for classifications problems. In order to model inhabitant's activities in 3D, The simulator was built using Unity3D (2005).

The intelligent environment simulation (IE Sim) (Synnott *et al.*, 2014) is a tool used to generate simulated datasets that capture normal and abnormal ADLs of inhabitants. It allows the researcher to design smart homes by providing a 2D graphical top-view of the floor plan. The researcher can add different types of sensors such as temperature sensors, pressure sensors, etc. Then, using an avatar, the simulation can be conducted to capture ADLs. The format of the generated dataset is homeML (McDonald *et al.*, 2013). To the knowledge of the author, IE Sim is not available in the public domain.

Ariani *et al.* (2013) developed a smart home simulation tool that uses ambient sensors to capture the interactions of the inhabitants. The tool has a map editor that allows the researcher to design a floor plan for a smart home by drawing shapes on a 2D canvas. Then, the researcher can add ambient sensors to the virtual home. The tool can simulate binary motion detectors and binary pressure sensors. To simulate the activities and interactions in the smart home, they used the A* pathfinding algorithm (Hart *et al.*, 1968), to simulate the movement of the inhabitants. During the simulation, all interactions are sampled at 5 Hz and stored into an XML file.

UbiREAL (Nishikawa *et al.*, 2006) is a Java-based simulation tool that allows the development of ubiquitous applications in a 3D virtual smart space. It allows the researcher to simulate the operations and communications of the smart devices at the network level.

V-PlaceSims (Lertlakkhanakul *et al.*, 2008) is a simulation tool that allows a smart home designer to design a smart home from a floor plan. Then, allows multiple users to interact with this environment through a web interface. The focus of this tool is the improvement of the designs and management of the smart home.

In addition to the outlined above simulation tools, there are other commercial simulation tools targeting the industry such as FlexSim (FlexSim Software Products, Inc., 1993), Simio (LLC, 2006) and Arena (Automation, 2000).

Generally, the model-based approach allows the researcher to generate large datasets in short simulation time but sacrifices the granularity of capturing realistic interactions. On the other hand, the interactive approach captures these realistic interactions but sacrifices the short and quick simulation time and therefore, the generated datasets are usually smaller than the ones generated by model-based approach.

4.2.3 Analysis

Synnott *et al.* (2015) identified several challenges that face the smart home simulation research. One of the key challenges is that many of the available simulation tools (Fu *et al.*, 2011; Stahl & Schwartz, 2010; Lertlakkhanakul *et al.*, 2008; Armac & Retkowitz, 2007; Park *et al.*, 2007; O'Neill *et al.*, 2005; Nishikawa *et al.*, 2006; Barton & Vijayaraghavan, 2002) focus on testing applications that provide context-awareness and visualisation rather than focusing on generating representative datasets. Few of the available tools do focus on generating datasets (Buchmayr *et al.*, 2011; McGlinn *et al.*, 2010; Poland *et al.*, 2009; Krzyska, 2006). Another key challenge is to have the flexibility and scalability to add new/customised types of smart devices, change their generated output(s), change their positions within the smart home, etc. The multiple inhabitants' support is also one of the limitations facing the currently available tools as this feature is known to be difficult to implement (Synnott *et al.*, 2015).

The review of available smart home simulation tools reveals that the majority of the reported work lacks the openness and availability of the software implementation, which hinders their benefit to the wider research community. Moreover, less than half of the reviewed tools (10 out of 23) do not support multiple operating systems which can be an issue when working with research teams and/or test subjects. Table 4.1 shows the analysis and comparison of the proposed tool, OpenSHS, with the existing simulation tools. SIMACT (Bouchard *et al.*, 2010) and UbiWise (Barton & Vijayaraghavan, 2002) were the only open-source and cross-platform simulation tools available, however, the data generation approach used in that tool is based on a pre-defined script that the researcher plays back within the 3D simulation view.

Apart from the work by Lundström *et al.* (2015), this analysis shows that none of the reviewed simulation tools follows a hybrid approach i.e. a tool that combines the ability of model-based tools to generate large datasets in a reasonable time while keeping the fine-grained interactions that are exhibited by the interactive tools.

This review shows that fewer simulation tools focus on generating datasets while the majority of the reviewed tools focus on visualisation and context-awareness applications.

Supporting the simulation of multiple inhabitants is a tricky task especially for the tools that focus on generating datasets. Most of these tools have an avatar controlled by a single participant at a given time. To allow multiple participants to conduct a simulation at the same time is one of the identified challenges.

TABLE 4.1: Analysis of smart home simulation tools.

Tool/Author(s)	Date	Open-source	3D	Cross-platform	Approach	Focus	Multi-inhabitants	Fast-forwarding
OpenSHS	**2017**	**Yes**	**Yes**	**Yes**	**Hybrid**	**Dataset generation**	**Partially**	**Yes**
Park *et al.* (2015)	2015	No	Yes	Yes	Interactive	Visualisation	No	Yes
PerSim 3D (Lee *et al.*, 2015)	2015	No	Yes	Yes	Model-based	Dataset generation	No	Not applicable
IE Sim extended (Lundström *et al.*, 2015)	2015	No	No	No	Hybrid	Dataset generation	No	Yes
IE Sim (Synnott *et al.*, 2014)	2014	No	No	No	Interactive	Dataset generation	No	No
Kormányos & Pataki (2013)	2013	No	No	No	Model-based	Visualisation	No	Not applicable
Ariani *et al.* (2013)	2013	No	No	No	Interactive	Dataset generation	Yes	No
Fu *et al.* (2011)	2011	No	No	Yes	Interactive	Visualisation	Yes	No
Jahromi *et al.* (2011)	2011	No	No	No	Model-based	Visualisation	No	Not applicable
Buchmayr *et al.* (2011)	2011	No	No	No	Interactive	Dataset generation	No	No
SimCon (McGlinn *et al.*, 2010)	2010	No	Yes	Yes	Interactive	Dataset generation	No	No
YAMAMOTO (Stahl & Schwartz, 2010)	2010	No	Yes	Not reported	Interactive	Visualisation	No	No
SIMACT (Bouchard *et al.*, 2010)	2010	Yes	Yes	Yes	Model-based	Visualisation	No	Not applicable
Poland *et al.* (2009)	2009	No	Yes	Yes	Interactive	Dataset generation	No	No
ISS (Van Nguyen *et al.*, 2009)	2009	No	No	No	Interactive	Visualisation	Yes	No
DiaSim (Jouve *et al.*, 2009)	2009	No	No	Yes	Model-based	Visualisation	No	Not applicable
V-PlaceSims (Lertlakkhanakul *et al.*, 2008)	2008	No	Yes	No	Interactive	Visualisation	Yes	No
Armac & Retkowitz (2007)	2007	Not reported	No	Not reported	Interactive	Visualisation	Yes	No
CASS (Park *et al.*, 2007)	2007	No	No	No	Model-based	Visualisation	Yes	Not applicable
Krzyska (2006)	2006	No	No	Yes	Interactive	Dataset generation	Yes	No
CAST (Kim *et al.*, 2006)	2006	No	No	No	Model-based	Visualisation	No	Not applicable
UbiREAL (Nishikawa *et al.*, 2006)	2006	No	No	Yes	Interactive	Visualisation	Yes	No
TATUS (O'Neill *et al.*, 2005)	2005	No	Yes	Not reported	Interactive	Visualisation	Yes	No
UbiWise (Barton & Vijayaraghavan, 2002)	2002	Yes	Yes	Yes	Interactive	Visualisation	Yes	No

When comparing OpenSHS against the available simulation tools reviewed in Table 4.1, unlike the majority of such tools, OpenSHS is based on Blender and Python which are open-source and cross-platform solutions, this offers the following benefits:

- Improves the quality of the state-of-the-art datasets by allowing the scientific community to openly converge on standard datasets for different domains,
- Allows easier collaborations between research teams from around the globe,
- Facilitates developments and lower entry barriers,
- Facilitates the objective evaluations and assessments.

OpenSHS allows the simulations to be conducted in 3D from a first-person perspective. The only open-source tools identified in the literature were SIMACT (Bouchard *et al.*, 2010) and (Barton & Vijayaraghavan, 2002). However, both of these tools are not focusing on generating datasets. SIMACT does not allow the participant to create specialised simulations. Instead, it relies on pre-recorded data captured from clinical trials.

IE Sim (Synnott *et al.*, 2014) was extended to use a probabilistic model (Poisson distribution) to augment the interactively recorded data by IE Sim. Therefore, the extended version of IE Sim uses a hybrid approach. However, IE Sim is a 2D simulator which takes part of the realism out of the simulation. This might be a problem when 3D motion data is important to the researcher, for example in anomaly detection algorithms, as identified by Lundström *et al.* (2015).

The fast-forwarding feature makes the simulation less cumbersome especially when the simulation has long periods of inactivity as in elder care research. This feature is relevant to interactive and hybrid approaches. OpenSHS's fast-forwarding mechanism streamlines the performance of the simulation and allows the participant to skip in time while conducting a simulation.

Although, OpenSHS currently supports the simulation of one smart home's inhabitant, however multiple inhabitants simulations are partially supported. The current implementation of this feature does not allow real-time simulation of multiple inhabitants. Instead, the first inhabitant records his/her activities and then the second inhabitant can start another simulation. The second inhabitant will be able to see the first inhabitant's actions played back in the virtual environment.

The approach that OpenSHS uses to generate datasets can be thought of as a middle ground between the model-based and interactive approaches. The replication mechanism that OpenSHS adapts, allows for a quick dataset generation, similar to the model-based approaches. Moreover, the replications have richer details as the activities are

captured in real-time, similar to the interactive approaches. Overall, the advantages of OpenSHS can be summarised as follows:

1. **Accessibility**: The underlying technologies used to develop OpenSHS allowed it to work on multiple platforms, thus ensuring a better accessibility for the researchers and the participants alike.
2. **Flexibility:** OpenSHS gives the researchers the flexibility to simulate different scenarios according to their needs, by adding and/or removing sensors and smart devices. OpenSHS can be easily modified and customised in terms of positioning and changing the behaviour of the smart devices in the virtual smart home to meet the needs of a research project.
3. **Interactivity:** Capturing the interactions between the participant and the smart home in OpenSHS was done in a real-time fashion which facilitates the generation of richer datasets.
4. **Scalability:** The proposed simulation tool is scalable and easily extensible to add new types of smart devices and sensors. OpenSHS has a library of smart devices that can be developed and updated as new types of smart devices become available.
5. **Reproducibility:** By being an open-source project, OpenSHS does have the advantage of facilitating reproducibility and allowing research teams to produce datasets to validate other research activities.

4.3 OpenSHS Architecture and Implementation

This thesis proposes a new hybrid, open-source, and cross-platform 3D smart home simulation tool for dataset generation, OpenSHS (Alshammari *et al.*, 2017a), which is downloadable from `http://www.openshs.org` under the GPLv2 licence (GNU, 1991). OpenSHS tries to provide a solution to the issues and challenges identified by Synnott *et al.* (2015). OpenSHS follows a hybrid approach, to generate datasets, combining the advantages of both model-based and interactive approaches. This Section presents the architecture of OpenSHS and the technical details of its implementation, which is based on Blender (1995) and Python. In this Section, two entities will be referred to, the researcher and the participant. The researcher is responsible for most of the work with OpenSHS. The participant is any person volunteering to simulate their activities.

Working with OpenSHS can be divided into three main phases: design phase, simulation phase, and aggregation phase. The following subsections will describe each phase.

4.3.1 Design Phase

In this phase, as shown in Figure 4.2, the researcher builds the virtual environment, imports the smart devices, assign activities' labels and design the contexts.

4.3.1.1 Designing Floor Plan

The researcher designs the 3D floor plan by using Blender which allows the researcher to easily model the house architecture and control different aspects such as the dimensions and the square footage. In this step, the number of rooms and the overall architecture of the home is defined according to the requirements of the experiment.

4.3.1.2 Importing Smart Devices

After the design of the floor plan, the smart devices can be imported into the smart home from the smart devices' library, offered by OpenSHS. The current version of OpenSHS includes the following list of active and passive devices/sensors:

- Pressure sensors (e.g. activated carpet, bed, couch, etc.),
- Door sensors,
- Lock devices,
- Appliance switches (TV, oven, fridge, etc.),
- Light controllers.

The Smart devices' library is designed to be a repository of different types of smart devices and sensors. This list is extensible as it is programmed with Python. Moreover, the researcher can build a customised sensor/device.

4.3.1.3 Assigning Activity Labels

OpenSHS enables the researcher to define an unlimited number of activities' labels. The researcher decides how many labels are needed according to their experiment's requirements. Figure 4.4 shows a prototype where the researcher identified five labels. Namely, 'sleep', 'eat', 'personal', 'work' and 'other'. This list of activity labels represents a sample of activities, which the researchers can tailor it to their needs.

4.3.1.4 Designing Contexts

After designing the smart home model, the researcher designs the contexts to be simulated. The contexts are specific time frames that the researcher is interested to simulate e.g. morning, afternoon, evening contexts. For instance, if the researcher aims to simulate the activities that a participant performs when he/she comes back from work during a weekday; then the researcher will design a context for that period. Finally, the researcher specifies the initial states of the devices for each context.

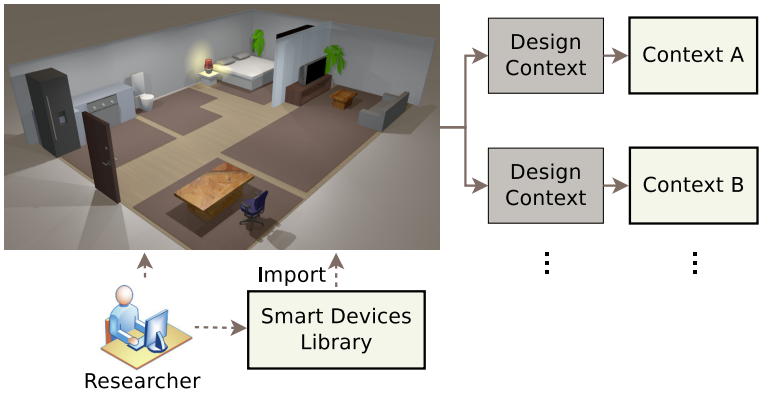


FIGURE 4.2: The design phase.

4.3.2 Simulation Phase

Figure 4.3 shows the overall architecture of the simulation phase. The researcher starts the tool from the OpenSHS interface module which allows the researcher to specify which context to simulate. Each context has a default starting date and time and the researcher can adjust the date and time if he/she wants. Every context has a default state for the sensors the 3D position of the avatar. Then, the participant starts simulating his/her ADLs in that context. During the simulation time, the sensors' outputs and the state of different devices are captured and stored in a temporary dataset. OpenSHS adapts a sampling rate of one second by default, which the researcher can re-configure as required. Once the participant finishes a simulation, the application control is sent back to the main module to start the simulation of another context.

The simulation phase aims to capture the granularity of the participants' realistic interactions. However, capturing these fine-grained activities in extended periods of time adds a burden on the participant(s) and sometimes becomes infeasible. OpenSHS offers a solution that mitigates this issue by adapting a fast-forwarding mechanism.

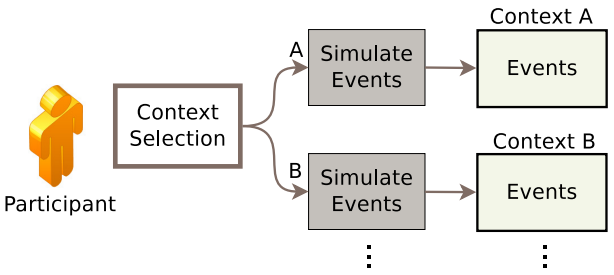


FIGURE 4.3: The simulation phase.

4.3.2.1 Fast-Forwarding

OpenSHS allows the participant to control the time span of a certain activity, fast-forwarding. For example, if the participant wants to watch the TV for a period and does not want to perform the whole activity in real-time (since there are no changes in the readings of the home's sensors), the participant can initiate that activity and spawn a dialogue to specify how long this activity lasts. This feature allows the simulation process to be quick and streamlined. The tool will simply copy and repeat the existing state of all sensors and devices during the specified time period. Figure Figure 4.4 shows the activity fast-forwarding dialogue during a simulation.

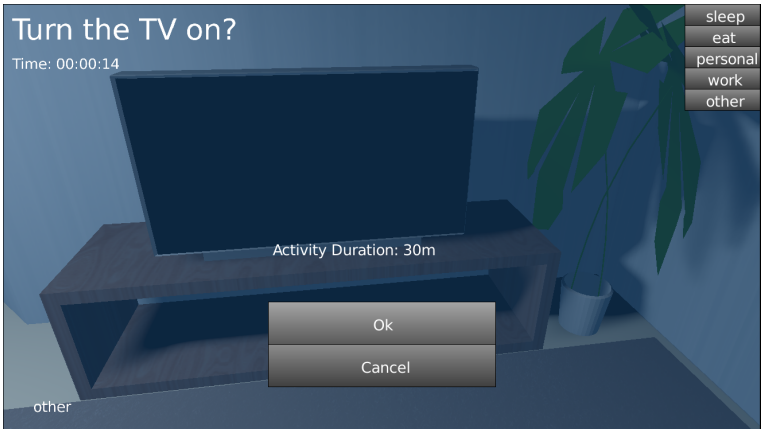


FIGURE 4.4: The activity selection and fast-forwarding dialogue.

4.3.2.2 Activities Labelling

The researcher is responsible for familiarising the participant with the available activity labels to choose from. During a simulation and before transitioning from one activity to another, the participant will spawn the activity dialogue shown in Figure 4.4 to choose the new activity from the available list. To ensure a clean transition from one activity to another, OpenSHS will not commit the new label at the exact moment of choosing.

Instead, the new label will be committed when a sensor changes its state. For example, in Figure 4.6 the transition from the first activity (sleep) to the second (personal) is committed to the dataset when the sensor `bedroomLight` changes its state even though the participant did change the label a couple of seconds earlier.

4.3.3 Aggregation Phase

After performing the simulation by the participants, the researcher can aggregate the participants' generated sample activities i.e. events, in order to produce the final dataset. The results of the simulation phase forms a pool of sample activities for each context. The aggregation phase aims to provide a solution for the generation of large datasets in short simulation time. Hence, this work develops an algorithm that replicates the output of the simulation phase by drawing appropriate samples for each designated context.

This feature encapsulates model-based approach advantage with the interactive approach adapted by the simulation phase, which allows OpenSHS to combine the benefits of both approaches, a hybrid approach.

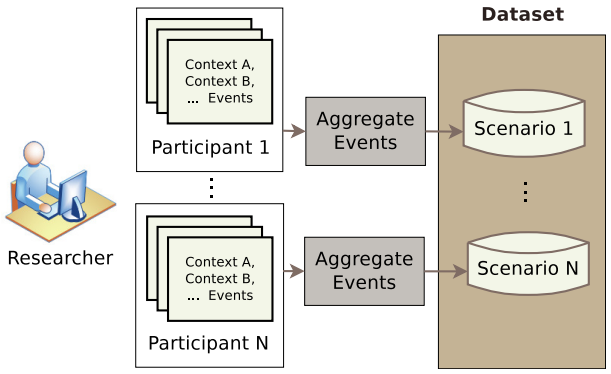


FIGURE 4.5: The aggregation phase.

4.3.3.1 Events Replication

It was evident from the beginning of the development of this simulation tool that it is not feasible for a participant to sit down and simulate his/her ADLs for a whole day. Moreover, capturing the interactions between the inhabitant and the smart home in real-time was needed. At the same time, the process should be less tedious and streamlined as much as possible. These requirements brought up the concept of real-time context simulations. Instead of having the user simulating his/her ADLs for extended periods of time, the user simulates only a particular context in real-time. For example, let us assume the researcher is interested in an 'early morning' context and wants to capture

the activities that the inhabitant is doing in this time frame, such as, what is usually done in the weekdays compared to the weekends in the same context (The 'early morning' context). The user will only perform sample simulations of different events in real-time. The more the number of samples simulated, the richer the generated dataset will be.

To gain more insight of how OpenSHS works, a virtual smart home environment consisting of a bedroom, a living room, a bathroom, a kitchen, and an office was built. Each room is equipped with several sensors totalling twenty-nine sensors of different types. The sensors are binary, and they are either on or off at any given time step.

The result of performing a context simulation can be illustrated by Figure 4.6. The sample consists of three activity labels, namely 'sleep', 'personal', and 'other'. Each activity label corresponds to a set of sensors' readings. The sensors' readings in the previous figure are readings of binary sensors and the small circles correspond to an 'ON-state' of that sensor.

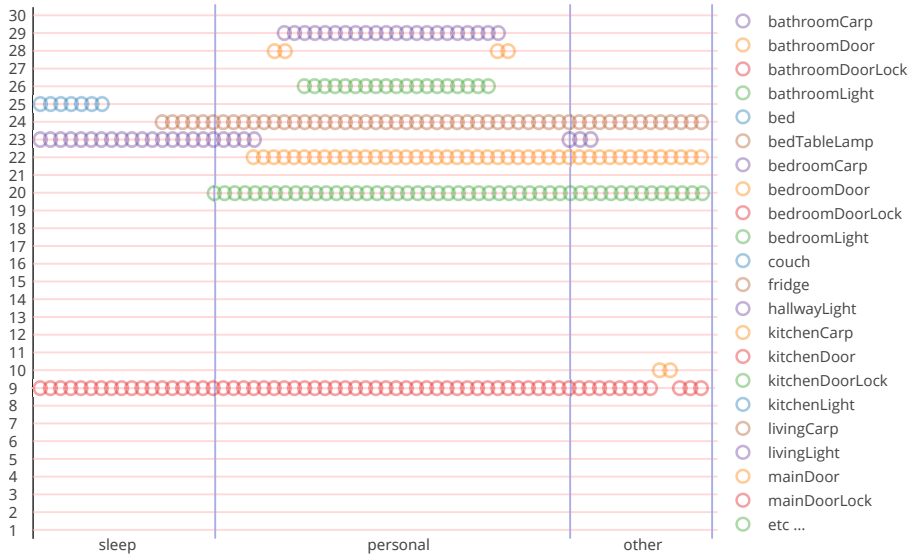


FIGURE 4.6: Twenty-nine binary sensors' output and the corresponding activity labels.

TABLE 4.2: A set of recorded samples for a particular context.

SAMPLE 1	sleep	personal	work	eat	other
SAMPLE 2	sleep	personal	other		
SAMPLE 3	sleep	personal	other		
SAMPLE 4	sleep	eat	personal	other	
SAMPLE 5	sleep	eat	personal	other	

It is not realistic to aggregate the final dataset by trivially duplicating the contexts samples. This will create exact duplicates of the samples. There is a need for an algorithm that can replicate the recorded samples to generate a larger dataset. A replication algorithm for extending and expanding the recorded samples was designed. A small number of simulated events can be extended without affecting their logical order.

To explain the replication algorithm, it is best illustrated by an example. Table 4.2 shows a set of five samples with their activity labels for a certain context. The first sample has five activities and the second sample has three activities and so on. When the researcher aggregates the final dataset, the samples of every context are grouped by the number of activities in each sample. So for the previous example, sample 1 will be in one group, sample 2 and 3 will be in a second group, and sample 4 and 5 will be in a third group. Then, a random group will be chosen and from that group, a sample will be drawn for each activity. For example, let us take the second group which contains sample 2 and 3. The number of activities in this group is three. So, for the first activity, the algorithm will either pick the 'sleep' activity from sample 2 or the 'sleep' activity from sample 3. The same procedure will be done for the second and third activities. The output will resemble what is shown in Table 4.3.

TABLE 4.3: Ten replicated copies based on the samples from Table 4.2.

i	Activity 1	Activity 2	Activity 3	Activity 4	Activity 5
1.	**sample 1** sleep	**sample 1** personal	**sample 1** work	**sample 1** eat	**sample 1** other
2.	**sample 4** sleep	**sample 5** eat	**sample 5** personal	**sample 4** other	
3.	**sample 3** sleep	**sample 3** personal	**sample 2** other		
4.	**sample 3** sleep	**sample 3** personal	**sample 2** other		
5.	**sample 5** sleep	**sample 4** eat	**sample 5** personal	**sample 5** other	
6.	**sample 1** sleep	**sample 1** personal	**sample 1** work	**sample 1** eat	**sample 1** other
7.	**sample 2** sleep	**sample 2** personal	**sample 2** other		
8.	**sample 5** sleep	**sample 5** eat	**sample 5** personal	**sample 5** other	
9.	**sample 4** sleep	**sample 4** eat	**sample 4** personal	**sample 5** other	
10.	**sample 2** sleep	**sample 2** personal	**sample 2** other		

The context samples shown in Table 4.2 will produce 25 unique replicated copies. In general, the number of unique replicated copies for a single context can be calculated by the Equation (4.1). Let $\mathcal{G}$ denotes the number of the groups of unique length of activities, and let $\mathcal{S}_g$ denotes the number of samples for the group g, and let $\mathcal{A}$ denotes the number of activities within a sample $\mathcal{S}_g$. The total number of unique replicated copies $\mathcal{R}$ is:

$$\mathcal{R} = \sum_{g=1}^{\mathcal{G}} \mathcal{S}_g^{\mathcal{A}} \tag{4.1}$$

OpenSHS can modify the original duration of a performed activity by shortening and/or expanding it. To preserve the structure of a certain activity, the algorithm looks for the longest steady and unchanged sequence of readings. Then, the proposed algorithm randomly chooses a new duration for this sequence. The new modified sequence length can vary between 5% of the original sequence length, up to its full length. The researcher can use this feature by passing the `variable-activities` option to the aggregation parameters as will be shown next.

The researcher can configure a number of parameters to control the generated output such as:

- `days`: the number of days to be generated,
- `start-date`: specifies the starting date for the dataset,
- `time-margin`: the variability of the starting time for the replicated events. For example, assuming a sample that was recorded at 7:30 am and the time margin specified to be 10 minutes. The replicated sample could start anytime from 7:25 am up to 7:35 am,
- `variable-activities`: make the duration for each activity variable.

4.3.3.2 Dataset Generation

After running the aggregation algorithm, the researcher can combine all the scenarios, generated by different participants, into one final comma separated values (CSV) dataset output. Table 4.4 shows a sample generated dataset.

TABLE 4.4: A sample of the final dataset output.

timestamp	bedTableLamp	bed	bathroomLight	bathroomDoor	...	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
⋮	⋮	⋮	⋮	⋮	⋮	

The `time-margin` parameter does add a level of sophistication to the timing of the recorded activities. This is useful for applications that rely heavily on the time dimension of activities, for example, in anomaly detection research.

4.3.4 Implementation

OpenSHS implementation relies on Blender and its game engine. Blender's game engine is programmable by Python.

4.3.4.1 Blender

Blender was chosen to build the majority of the simulation tool and to act as an infrastructure for OpenSHS. The reasons for this choice can be summarised as:

- **Open-source:** Blender is an open-source 3D modelling and animation software and an actively developed project by the open-source community. It allows the user to create 3D models and visual effects. The Game Engine component of Blender allows the user to build complex 3D interactive games and script them with Python which is an important feature for OpenSHS.
- **Cross-platform**: Blender is available for the three major operating systems. Namely, GNU/Linux, Microsoft Windows, and Apple macOS. Blender uses OpenGL (1992) for its Game Engine which is also, a cross-platform 3D technology available for the major operating systems.

- **The Blender Game Engine**: Blender's Game Engine allowed us to add the interactivity to the simulations. The physics engine facilitates the simulation of different types of real sensors and devices. For example, blender has a 'Near' sensor which will only be activated when the 3D avatar controlled by the user is physically near other objects in the scene. Therefore, such sensor could be used to simulate a proximity sensor easily.

4.3.4.2 Python

The interaction with the simulation tool is done by controlling a 3D avatar that navigates the smart home space through a first-person perspective similar to most first-person games. Figure 4.7 shows the 3D avatar navigating the living room. Since Blender's Game Engine uses Python as a programming language, all the logic and interactions between the avatar and the virtual environment was developed with it. Moreover, all of OpenSHS modules are programmed by Python.

FIGURE 4.7: Navigating the smart home space through the first-person perspective.

4.4 OpenSHS Usability

Measuring the usability of a software tool is a challenging and tricky task since it involves subjective qualities and depends on the context used. Brooke *et al.* (1996) defines it as *"The general quality of the appropriateness to a purpose of any particular artefact"*. He developed the widely used System Usability Scale (SUS) which is a questionnaire consisting of ten questions that measure various aspects of the usability of a system. The score of SUS ranges from 0 to 100.

To assess OpenSHS usability, a usability study using SUS was conducted. Our sample consists of graduate students and researchers interested in smart home research. Multiple sessions were carried out and each session started by introducing OpenSHS and then by presenting its functionalities. After that, any questions the participants had in mind were answered. Afterwards, the participants were allowed to use OpenSHS and explore its features. Finally, the participants were asked to answer few questions, such as how frequently do they use their computer on daily basis and whether they play first-person 3D video games or not. Then, the participants were asked to fill the SUS questionnaire.

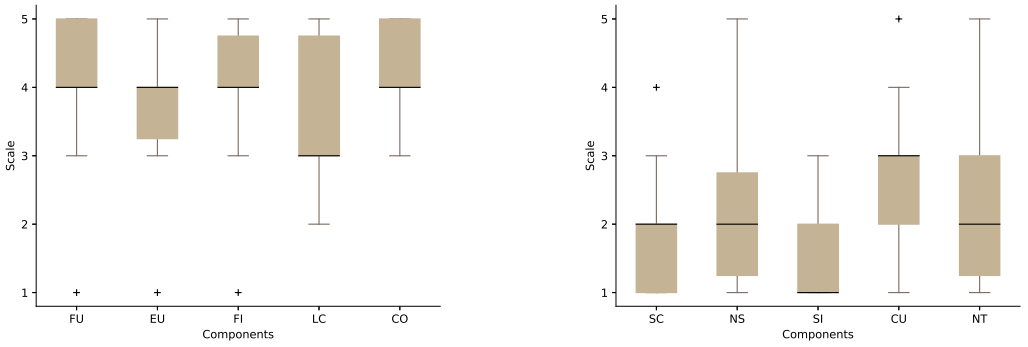
Two usability studies were carried out. One form the perspective of the researchers and the other from the perspective of the participants using OpenSHS. The researchers' group were asked to evaluate OpenSHS usability throughout the three phases (design, simulation, aggregation). The participants group were only requested to evaluate the simulation phase.

For the researchers' group, data from 14 researchers were collected, 85.7% were male, and 14.3% female. The average age of the researchers was 36 ($min_{age} = 31, max_{age} = 43$). All the researchers reported that they do use their computers on a daily basis and 93% of them did play 3D first-person games. The aspects that the SUS questionnaire investigates can be summarised as:

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.

Figure 4.8 shows the results of our SUS questionnaire for the researchers' group. The odd-numbered statements contributes positively to the overall score if the participant agrees with them (Figure 4.8a). On the other hand, the even-numbered statements contributes negatively if the researcher agrees with them (Figure 4.8b). Calculating the score of our sample revealed that the average SUS score of OpenSHS is 71.25 out of 100 ($score_{min} = 40, score_{max} = 85$).

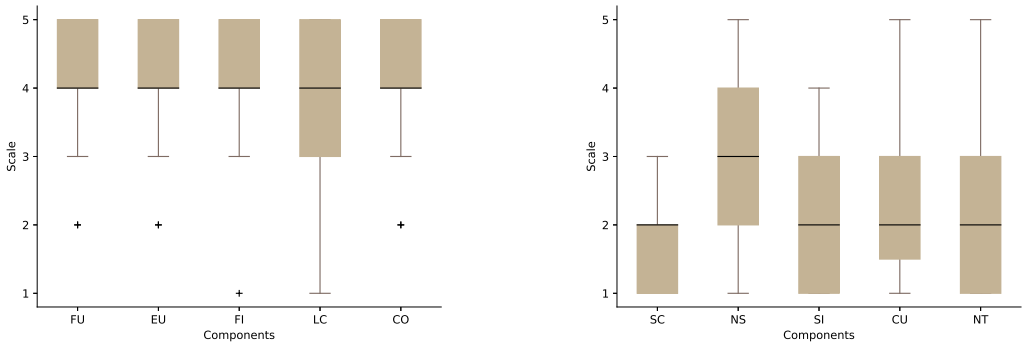


(A) The positive components. (B) The negative components.

FIGURE 4.8: The result of System Usability Scale (SUS) questionnaire for the researchers' group.

For the participants' group, 31 participants were asked to answer the SUS questionnaire. 77.5% were male, and 22.5% female and average age of the participants was 27 ($min_{age} = 21, max_{age} = 36$). 97% did play first-person games and all of the participants reported that they use their computers on daily basis. Figure 4.9 shows the participants' group results. The SUS score for this group is 72.66 out of 100 ($score_{min} = 50, score_{max} = 87$).

The usability results for both groups are promising but, at the same time, they indicate that there is room for improvements. Both groups agree that the learning curve (LC) component of the questionnaire needs improvement. The results also show the need for support from a technical person to use the system.



(A) The positive components.

(B) The negative components.

FIGURE 4.9: The result of System Usability Scale (SUS) questionnaire for the participants group.

4.5 Conclusion

Many smart home research projects require the existence of representative datasets for their respective applications and research interests and to evaluate and validate their results. Many simulation tools available in the literature focus on context-awareness and few tools have set dataset generation as their aim. Moreover, there is a lack of open-source simulation tools in the public domain. OpenSHS, an open-source, 3D and cross-platform simulation tool for smart home dataset generation was developed. OpenSHS has many features that allow the researchers to easily design different scenarios and produce highly intricate and representative datasets. Our tool offers a library of smart sensors and devices that can be expanded to include future emerging technologies.

OpenSHS allows the researchers to generate seeds of events rapidly. A replication algorithm that can extend the simulated events to generate multiple unique large datasets was introduced. Moreover, conducting a simulation with a participant can be done in a reasonable time and the tools that streamline the process such as fast-forwarding were provided.

The proposed tool divides the dataset generation process into three distinct phases, design, simulation and aggregation. In the design phase, the researcher creates the initial virtual environment by building the home, importing smart devices and creating contexts. In the simulation phase, the participant uses the virtual home to generate context-specific events. In the final stage, the researcher applies the replication algorithm to generate the aggregated dataset.

A usability study using the System Usability Scale (SUS) to assess how usable OpenSHS was conducted. The results of this study were promising, yet they left room for more improvements.

One of the identified issues in smart home simulations tools, is having the support for multiple inhabitants. This is a challenging task both for the simulation tool and for the participants. Currently, OpenSHS offers partial support for multiple inhabitants. To increase the realism of the simulations, integrating VR technologies into OpenSHS in the future is planned. The accessibility for both the researchers and the participants is an important feature. Hence, a port of the implementation of OpenSHS to run in a web browser is planned.

Chapter 5

HI-SDR Encoder

5.1 Introduction

All HTM systems have an encoding region that converts the raw input data to an SDR, the basic data structure of any HTM system. Due to the nature of the input data, whether numerical, categorical, single-column or multi-columnar, the encoder's job is to convert this input in a way that allows the HTM system to learn and recognise the patterns. Having good encoders is a crucial requirement for the whole system to perform well. These play the role of our senses that translate what we see, hear or touch to a representation that our brains can process.

This Chapter proposes a novel encoder and starts by presenting a formal definition of SDRs and the mathematical notations used in this work and in the HTM literature in general. The Chapter presents what makes an encoder produce good SDRs and the properties which are required for this. The standard NuPIC encoders and a study of their behaviour will be presented. The issue with the standard NuPIC encoders when dealing with multi-columnar datasets as seen in the smart home datasets will be explored. Several experiments will be conducted to highlight these issues and analyse the encoders' performance. The Chapter will conclude with the novel Hash Indexed Sparse Distributed Representation encoder (HI-SDR) that does resolve these issues.

5.2 Sparse Distributed Representations

The SDR is the basic information representation and one of the main building blocks in any HTM system. This Section presents a mathematical formalisation of the SDRs and the basic operations that can be performed on them.

5.2.1 Notations

The notations and definitions used in this Section is based on the work presented by Ahmad & Hawkins (2015). The following listing is a set of definitions and mathematical notations that will be used throughout this work:

SDR: is a one-dimensional binary array consisting of mostly zeros and very small percentage of ones (active bits). The active bits usually constitute around 2%. The total number of bits in an SDR is referred to by n. An SDR x has n-length array of binary components b_i where i is the index of the component in the array. e.g. $x = [b_0, b_1, \ldots, b_{n-1}]$. The total number of components with the value 1 for the SDR x is denoted by w_x. The variable w was chosen as a shorthand for $width$. The 1-valued components are referred to as *active bits* since they represent a firing/active neuron. Here is an example of an SDR x with $n = 10$ and $w = 3$:

$$x = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

The set of active bits w does not have to be consecutive. In the following sections different types of SDRs will be presented where the active bits are distributed across the available space.

Overlap: One of the primitive operations done on SDRs is comparing the similarity between two SDRs and is referred to this operation as the overlap score. The name 'overlap' was chosen because when two SDRs are overlaid on top of each other, the similarity between them is determined by the number of active bits that are in the same location for both SDRs. Thus, the similarity score equals to the number of bits overlapping between any two SDRs. The overlap score can also be expressed as the dot product of two SDRs, for example, *x* and *y* as such:

$$overlap(x, y) = x \cdot y \tag{5.2}$$

Match: The notion of having two SDRs matching each other can be realised with the aid of the previously mentioned 'overlap' function. We introduce a variable θ to denote the threshold which determines if there is a match or not for any two given SDRs. Therefore, a 'match' can be expressed by:

$$match(x, y) = overlap(x, y) \geq \theta \tag{5.3}$$

The variable θ is less than or equal to the active bits for any given SDR ($\theta \leq w$).

Sparsity: is the ratio of the active bits w to the total number of bits n. This function is denoted by s. For example, for an SDR x, the sparsity of this SDR will be:

$$s(x) = \frac{w}{n} \tag{5.4}$$

5.2.2 SDRs Properties

The number of unique SDRs that can be generated under different parameters can be formalised by:

$$\binom{n}{w} = \frac{n!}{w!(n-w)!} \tag{5.5}$$

The numerator of equation 5.5 represents the factorial of the total number of bits whether being active or inactive. The denominator is the factorial of the number of active bits multiplied by the number of inactive bits.

To take a simple case, assuming $n = 16, w = 1$. This results in 16 unique SDRs that can be used. By increasing w to 2, the capacity will increase to be 120. That is because the two active bits can be arranged in more unique ways in the available space.

When plugging in the values $n = 40, w = 4$, the number of unique SDRs that can be generated will be 91390. This number might seems small when compared against a dense binary representation such as ASCII coding which will generate 2^n unique representations. However, practically speaking, HTM encoders usually have $n = 2048, w = 40$ which gives 2.37×10^{84} unique representations (Ahmad & Hawkins, 2015).

To calculate the probability of having two SDRs (x, y) being identical, based on the same n and w:

$$P(x = y) = \frac{1}{\binom{n}{w}} \tag{5.6}$$

To explore the past two equations with concrete numbers, let us assume $n = 1024, w = 2$, this will produce 523776 unique representations and the probability of having two SDRs being identical will be 0.000001909. The probability will continue to decrease as the value of w increases to a point where number reaches a global minimum. Past that point and as w increases, the probability will increase. Therefore, there is an optimal value where the chance of having two identical SDRs is at its global minimum.

In the HTM literature there is the idea of *overlap set* which is a set of SDRs with the same n and w and matching a certain overlap score. This idea allows us to explore some mathematical properties of SDRs when doing more complex operations on groups of SDRs. The number of the members of the overlap set Ω for the SDR x with the parameters n and w under the condition of having exactly b overlapping bits between the SDR x and the members of the overlap set is:

$$|\Omega_x(n, w, b)| = \binom{w_x}{b} \times \binom{n - w_x}{w - b} \tag{5.7}$$

This is under the assumptions that $b \leq w_x$ and $b \leq w$. The first term in equation 5.7 is the number of subsets of SDR x with b active bits. The second term is the number of other SDRs with $n - w_x$ total bits and $w - b$ active bits. Let us see the simplest case possible, assuming $n = 600, w_x = 40, w = 40, b = 40$, there is only one SDR that satisfy these conditions. If a little more relaxed condition $n = 600, w_x = 40, w = 40, b = 39$ was taken, this will give us 22400 SDRs that fall into this overlap set.

Practically, the HTM system does not require to have exact matches where $\theta = w$. This is to combat the noise in the input signal. There is a trade off between increasing θ value and the system tolerance and sensitivity. Increasing θ will increase the system sensitivity and vice versa. For instance, let us assume an SDR x and a copy of it with 50% noise x'. If $w = 40$ and $\theta = 20$, then the system will still consider x and x' the same input. However, lowering θ will make the system consider a signal identical to another falsely, increasing the false positives. (Ahmad & Hawkins, 2015).

To calculate the probability of false positives in an HTM system under certain conditions, that is, the probability of $overlap(x, y) \geq \theta$ will be:

$$fp_w^n(\theta) = \frac{\sum_{b=\theta}^{w} |\Omega_x(n, w, b)|}{\binom{n}{w}} \tag{5.8}$$

Taking the previous parameters $n = 600, w_x = 40, w = 40, b = 39$, the chance of having a false positive match is very low (5.17×10^{-59}). Lowering $b = 20$, the chance is still very low (8.53×10^{-16}). Taking it to the lowest $\theta = 1$, the chance of false positive is 18%.

The SDRs, as shown earlier, have good noise robustness. This robustness can be exploited to achieve a couple of interesting capabilities. Using just a subset of the active bits when performing a matching operation and reliably know that the match is correct and not a false positive. To formalise this idea, let us take an SDR x and a sub-sampled

copy of it x' with $w_{x'} \leq w_x$. To calculate the probability of falsely matching the sub-sampled SDR x' with another random SDR y the overlap set of x' has to be calculated as follows:

$$|\Omega_{x'}(n, w_y, b)| = \binom{w_{x'}}{b} \times \binom{n - w_{x'}}{w_y - b} \tag{5.9}$$

The probability of a false positive will be:

$$fp^n_{w_y}(\theta) = \frac{\sum_{b=\theta}^{w_{x'}} |\Omega_{x'}(n, w_y, b)|}{\binom{n}{w_y}} \tag{5.10}$$

To plug concrete numbers in the previous equation, assuming $n = 1024$ for all the SDRs in this example. The original SDR x has $w_x = 8$, the sub-sampled SDR x' has $w_y = 4$, and a random SDR y has $w_y = 8$. So our sub-sampling ratio is 50%. And let us assume that the condition for a match $\theta = 2$. The probability of a false positive will be 0.000317. When increasing $\theta = 3$, the probability will be 0.00000125. Increasing it to the maximum allowed value under our assumptions $\theta = 4$ will give us a small chance of having false positive match (1.54×10^{-9}).

One of the properties of SDRs is that an SDR can be recognised and identified correctly from a group of SDRs. Let us assume X to be a set of unique M SDRs, $X = x_1, x_2, \ldots, x_m$ where every SDR in that set is different from all the other SDRs in the same set, that is:

$$\forall_{x \in X} \forall_{y \in X, y \notin x} match(x, y) = false \tag{5.11}$$

A random SDR y is said to belong to this set when:

$$y \in X \equiv \exists_{x_i \in X} match(x_i, y) = true \tag{5.12}$$

Now, given a noisy SDR x_i from the set X, and x_i has t bits of noise, under the assumption that $t \leq (w - \theta)$, the probability of falsely recognising it out of all the SDRs in the set X can be given by the following inequality that gives us an upper bound to this probability:

$$fp_x(t) \leq \sum_{i=1}^{M} fp^n_{w_{x_i}}(t) \tag{5.13}$$

If all the SDRs in the set X are of the same w, the upper bound will simply be:

$$fp_x(\theta) \leq M fp_w^n(\theta) \tag{5.14}$$

Assuming a set of 100 SDRs all of them are $n = 256, w = 4, \theta = 4$, The probability of falsely recognising one SDR in that set with another SDR in the same set is 5.72×10^{-7}. In a typical HTM system the parameters are usually in the range of $n = 2048, w = 40, \theta = 30$, which gives a probability of a false positive = 1.03×10^{-47}.

Therefore, a set of SDRs can be stored and retrieved back with high confidence. Again, this shows how the SDRs are noise tolerant if the parameters are set to a reasonable range.

There is another important property of SDRs that is commonly referred to as the 'union property' in the HTM literature. It is possible to store and merge many SDRs into one representation by performing a logical `OR` operation on all the SDRs. Because of the sparsity and the previously mentioned properties of the SDRs, it is possible to reliably store these SDRs without corrupting any of them, provided that there is sufficiently large bits. Here is an example of a union set of two SDRs of $n = 10, w = 1$:

$$\begin{aligned} x_1 &= [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \\ x_2 &= [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \end{aligned}$$

They can be merged into one SDR by applying the logical `OR` operation. The resulting SDR X will be:

$$X = [0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$

The sparsity of x_1 and x_2 is 1% and the resulting SDR X sparsity is 2%. In this simple example, it is possible to correctly know if a random SDR y is a member of this union set or not. i.e. is $match(X, y)$ true or not. From this example it might seem that the sparsity of the merged set is a linear function with respect to the number of the members in union set. However, This is not the case when having more typical encoder's parameters (see Figure 5.1).

The example above omits an important issue, what if there are collisions? In other words, what is the probability of getting false positives when matching with the union set? By starting with the simpler case, looking for exact matches i.e. when $\theta = w$, meaning all active bits in our query SDR must match with our union set. Having M SDRs in union set X, when $M = 1$, the probability of any bit to be 0 is $1 - \frac{w}{n}$. For any M SDRs, the probability is:

$$p_0 = (1 - \frac{w}{n})^M \tag{5.15}$$

After merging M SDRs into our union set X, the probability of any bit being active, in the resulting union set, is $1 - p_0$. Now the probability of a false positive where all the bits in our query SDR y matching our union set X is:

$$\begin{aligned} p_{fp} &= (1 - p_0)^w \\ &= (1 - (1 - \frac{w}{n})^M)^w \end{aligned} \tag{5.16}$$

Considering 10 SDRs with $n = 600, w = 2$, the probability of false positive is 0.0011. If w is increased to be 4, the probability of false positive rapidly goes down to be 1.75×10^{-5}. Increasing w yet again to be 6 gives a probability of 7.64×10^{-7}. Taking more typical parameters $n = 2048, w = 40$ and 20 SDRs in the union set, the probability of false positive is 3.37×10^{-20} which is extremely low.

The more SDRs added in the union set, the more likely that there will be a false positive. Now the question is how quickly does the function of the number of SDRs in the union set increases? The number of expected active bits in the union set is $n \times (1 - p_0)$. This function increases slower than a linear function after a certain threshold as shown in figure 5.1.

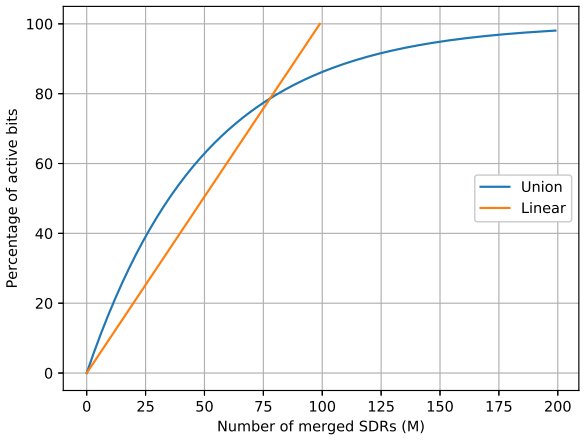


FIGURE 5.1: The growth of the number of active bits in the union set as a function of the number of SDRs in the set.

The previous analysis of the union set assumed that $\theta = w$, meaning that we are looking for exact and free of noise matching. Now let us explore the common case of not

requiring exact matching i.e. $\theta < w$. In this case, the expected number of active bits in the union set X is $\tilde{w_x} = n(1 - p_0)$ under the assumption that $n \geq \tilde{w_x} \geq w$, the overlap set size will be:

$$E[|\Omega_X(n, w, b)|] = \binom{\tilde{w_X}}{b} \times \binom{n - \tilde{w_X}}{w - b} \tag{5.17}$$

For a match to be true, there have to be an overlap of θ bits or more but not exceeding w. Therefore, the probability of false positives is approximately:

$$\epsilon \approx \frac{\sum_{b=\theta}^{w} |\Omega_X(n, w, b)|}{\binom{n}{w}} \tag{5.18}$$

Practically, in the equation 5.18, the first term of the sum does dominate. Thus, the previous equation can be simplified to:

$$\epsilon \approx \frac{|\Omega_X(n, w, b)|}{\binom{n}{w}} \tag{5.19}$$

Lowering the threshold θ will increase the probability of a false positive (Ahmad & Hawkins, 2016). However, increasing n will lower the chance of false positive and this is one of the trade offs in an HTM system (Ahmad & Hawkins, 2015).

All the operations on SDRs are performed on the active bits w. Therefore, the time complexity of the operations is $O(w)$ and the size of all the bits in an SDR n does not influence these calculations. Since the SDRs are sparse, in practice the w is much smaller than the overall number of bits n.

5.3 NuPIC Encoders

5.3.1 Proprieties of Good Encoders

According to Purdy (2016), any HTM system to function well and to produce good results it must have the following properties:

1. **Deterministic:** Given an input, the resulting SDR should always be the same when given the same input again.
2. **Fixed in dimensions:** The resulting SDRs should always have a fixed number of total bits.

3. **Fixed in sparsity:** The resulting SDRs should have fixed number of active bits.
4. **Capturing semantics:** Any two similar inputs should have an overlapping set of active bits.

Having a deterministic encoder is an important property for a good encoder and without it any HTM system will not function properly. When an HTM system is learning a sequence of SDRs and if the representations of the original values is changing over time, this will cause the HTM system not to recognise these SDRs and henceforth learn their succession over time.

The dimensionality of the encoder's output should be preserved throughout the life span of the learning process. This requirement becomes apparent when we realise that many of the primitive operations that take place in the SP and in the TM actually rely on bit-wise comparisons between a succession of SDRs. When the dimensionality of these SDRs is changing over time, this will lead to false calculations by the HTM system's components. Therefore, it is important to have the total number of bits fixed.

Similarly, the sparsity or the ratio of the active bits to the total number of bits, should be also fixed. Again, this is due to how the HTM system's primitive operations are calculated. How sparse should an encoder be? Is an open question and it might be application specific but generally it can range from 1% to 35% as suggested by Purdy (2016).

Defining what exactly is meant when saying that two inputs are semantically similar is a difficult task and it is highly dependent on the data type in question. Taking the natural numbers, for example, it is fairly easy to define the notion of semantic similarity between two natural numbers in a dataset if the value space for these numbers is known. By 'value space' the author means the range of values present in a dataset from a minimum known value to a maximum known value. For instance, taking the numbers **1** and **2** the minimum value allowed is **1** and the maximum value allowed is **100**, it can be said that the numbers **1** and **2** are semantically similar in this value space. And the numbers **1** and **100** are semantically the most dissimilar.

Purdy (2016) presented a formal mathematical description of the encoding procedure of semantically similar input and the number of overlapping active bits. This mathematical formalisation is as follows:

Let A be an input space and let $S(n, k)$ be the set of SDRs of length n with k active bits. An encoder f is a function that $A \rightarrow S(n, k)$. Let a distance score d_A be a function that $A \times A \rightarrow \mathbb{R}$ and satisfy the following conditions:

$$\forall x, y \in A, d_A(x, y) \geq 0 \tag{5.20}$$

$$\forall x, y \in A, d_A(x, y) = d_A(y, x) \tag{5.21}$$

$$\forall x \in A, d_A(x, x) = 0 \tag{5.22}$$

Equation (5.20) simply states that the similarity distance function should be positive or zero and equation (5.21) states that the distance function is symmetric and equation (5.22) states that for the same input, the distance is zero.

Moreover, Purdy (2016) provides a mathematical way of evaluating an encoder by comparing the distance scores with the number of overlapping active bits. Taking two inputs with a low distance score, the number of the overlapping active bits should be high. This evaluation procedure is formalised as:

For s and t being two resulting SDRs from an encoder, let $O(s, t)$ be the number of overlapping active bits of the two SDRs s and t. Then, an encoder $f : A \rightarrow S(n, k)$ and $\forall w, x, y, z \in A$:

$$O(f(w), f(x)) \geq O(f(y), f(z)) \Leftrightarrow d_A(w, x) \leq d_A(y, z) \tag{5.23}$$

5.3.2 Standard NuPIC Encoders

At the time of writing NuPIC has several built-in encoders for various data types. It is possible to divide the data types into two big categories: Numerical and Categorical data. In this Section, the currently available encoders in the NuPIC project and how they deal with the different data types will be presented.

5.3.2.1 Numerical Data Types

NuPIC has several encoders that work well with numerical data types. Figure 5.2 shows the Scalar encoder which is used to encode any scalar value. For each input, the resulting SDR is depicted as grid of dark (active bits) and light (inactive bits) squares. This 2D depiction is just for illustration purposes, while in reality, an SDR is just a 1D array. For this encoder to work it requires the user to specify the minimum and maximum values

for the data to be encoded. Also, it requires the total number of bits to be specified. These parameters have direct influence on the semantics of the data. In Figure 5.2, the minimum value to be specified is **1**, the maximum value is **100**, the total number of bits to be **100**, and the width (denoted by w) to be **3**. The *width* parameter is the number of active bits for any given input. Given the constraints imposed by these parameters, the number **1** and the number **2** share two overlapping bits, namely the second bit and the third bit. The number **1** and the number **3** only share one bit, namely the third bit. This configuration allows the HTM system to recognise that the number one and two are closely related than the number one and the number three. On the other hand, taking the number **100** it is possible to see that it shares no overlapping bits with any of the previously mentioned inputs.

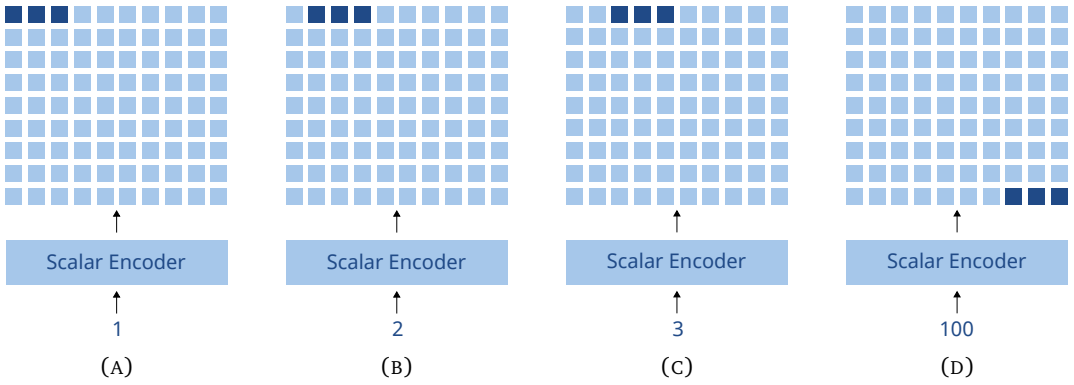


FIGURE 5.2: Encoding multiple scalar values $[1, 2, 3, 100]$ with the parameters $n = 100, w = 3, minVal = 1, maxVal = 100$.

In the HTM encoders literature there is this notion of *buckets*. The buckets are another way to specify the total number of bits (n) for a given scalar encoder. The number of buckets can be calculated as: $n - w + 1$. Therefore, the buckets can be thought of as the set of all the unique SDRs that can be produced given specific encoder parameters. It might be easier to explore this idea by an example. Assuming an encoder with the following parameters: $n = 10, w = 3, minVal = 1, maxVal = 10$, here are all the values from 1 to 10 and their resulting SDRs that this encoder can produce are:

```
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]
```

Every number in the previous example had a unique SDR except the numbers (3 & 4) and (7 & 8) each had the same SDR. When calculating the number of unique SDRs, it will be **8** which is also the number of buckets $(10 - 3 + 1)$.

The Scalar encoder interface provides several ways to specify the total number of bits so that the encoder will reflect the desired semantic properties for a given dataset. One of these alternative interfaces to the parameter n is the *resolution* parameter. Let us take an encoder with the following parameters: `resolution=0.5, w=3, minVal=1, maxVal=5`, it will be able to uniquely encode the following values:

```
1.0  →  [1 1 1 0 0 0 0 0 0 0 0]
1.5  →  [0 1 1 1 0 0 0 0 0 0 0]
2.0  →  [0 0 1 1 1 0 0 0 0 0 0]
2.5  →  [0 0 0 1 1 1 0 0 0 0 0]
3.0  →  [0 0 0 0 1 1 1 0 0 0 0]
3.5  →  [0 0 0 0 0 1 1 1 0 0 0]
4.0  →  [0 0 0 0 0 0 1 1 1 0 0]
4.5  →  [0 0 0 0 0 0 0 1 1 1 0]
5.0  →  [0 0 0 0 0 0 0 0 1 1 1]
```

From the previous listing it can be seen that every half a step in the range of values from 1 to 5, there is a unique SDR for that value. Therefore, the *resolution* parameter can be useful if the numerical values in a dataset are increasing by a fixed interval.

The Scalar encoder suffers form an issue that does not make it suitable to all scalar data. The encoder requires the minimum and maximum value to be defined before the user is able to encode the data with it. Moreover, the minimum and maximum values are set from the get-go, and they cannot be changed during the lifetime of the learning process. This limitation is inspired by biological sensors as they have fixed

range of values they are sensitive to. An example of this is the human eye sensitivity to the electromagnetic spectrum or the visible light. The human eye is only sensitive to a fraction of that spectrum ranging approximately from 380 nm to 800 nm. However, NuPIC does provide several other numerical encoders that offer a solution to overcome this issue, the Adaptive Scalar encoder is one of these encoders.

The Adaptive Scalar encoder works by keeping track of the minimum and maximum values as they are fed to the encoder and then adapting to these changes. To explore this idea let us take an Adaptive Scalar encoder with the following parameters: $n = 10, w = 3$, and feed the encoder the scalar values 1, 2, . . . , 10 twice to see the changes to the resulting SDRs. Here are the results of the 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 of the same scalar 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]
```

In the first pass, it can be seen that the first value (The number **1**) considered to be the minimum value and the second value fed in (The number **2**) is considered to be the maximum value as it happened to be bigger than the first. From the second value up to

the last, the resulting SDRs are identical and the active bits are occupying the maximum allowed representation. The encoder produced the same representation at each step because it keeps changing the maximum value to be the currently fed-in value and since these scalar values are increasing at each step, each one is taking the maximum value representation. However, on the second pass, the SDRs are adapting. It can be seen that the values' representations are shifting and creating a sliding window of active bits that cover the range of the values seen so far. It is worth noting that the Adaptive Scalar encoder interface does not provide a *resolution* parameter.

NuPIC also provides another dynamic encoder called Random Distributed Scalar encoder which overcomes the issue of having to specify the minimum and maximum values upfront. It utilises a hashing function to uniquely represent scalar values. This encoder keeps track of every scalar input and assign it a unique representation and as the input space grows, it dynamically grows the representations given the available space set by the encoder parameters. The Random Distributed Scalar encoder keeps and maintains the semantics meanings of the input (the number of overlapping bits). The way it does that can be explained by the aid of an example. Let us take a Random Distributed Scalar encoder with the following parameters `resolution=1, w=3, n=33` and let us encode the following values:

```
1   →   [00000000100010000000000000000010]
2   →   [00000000101010000000000000000000]
3   →   [00000000101000000000000100000000]
10  →   [00000000000100000010000000010000]
99  →   [01000000000010001000000000000000]
```

As shown in the example above, the number **1** and **2** have unique distributed representations, but they share two active bits out of the total three bits. The same is true for the numbers **2** and **3**, they only differ in one bit. However, the larger the entered number is, the larger the differences in the resulting representation. The Random Distributed Scalar encoder uses a hashing function to randomise and distribute the active bits over the SDR space. This example might be restrictive as the number of bits (n) should be larger in practice (The default value for n is 400). It is worth noting that there is no need to specify the parameter n and only use the `resolution` parameter to initialise the encoder, but that was done just for the sake of simplicity and illustration of how this encoder works. There is one aspect of this encoder which is not present in the other encoders presented in this Chapter, it does keep an internal state of all the encoded values it sees. It does that because it needs to know the mapping between the scalar values and the buckets before creating a new representation. This state ensures that the

resulting SDR capture the semantic meanings of the entered values. On the other hand, if the number of the entered values is big, this will require more memory and slower execution times.

Another encoder called the Log encoder that can be used with numerical values that exhibit log-like properties. Taking a Log encoder with the following parameters: `n=10`, `w=3`, `minVal=1`, `maxVal=1000`, then the results of the following values are:

```
1     →  [1 1 1 0 0 0 0 0 0 0]
2     →  [0 1 1 1 0 0 0 0 0 0]
10    →  [0 0 1 1 1 0 0 0 0 0]
100   →  [0 0 0 0 0 1 1 1 0 0]
1000  →  [0 0 0 0 0 0 0 1 1 1]
```

Another numerical encoder available is the Delta encoder which could be useful when the user wants to capture a numerical aspect about the input data which is not the actual values themselves but rather the rate of change of these values, for example.

Some numerical encoders offer the option to encode periodic data. For instance the Scalar encoder has a boolean flag named *periodic* which allows the user to specify whether the nature of the encoded data is cyclical and reoccurring. An example of numerical cyclical data could be the time of day. Capturing this property of the input data will allow the HTM system to better understand and learn the data. Having this flag enabled, will cause the active bits (w) to wrap around the allocated output space. Let us take an encoder with the following parameters: $n = 10, w = 3, minVal = 1, maxVal = 10, periodic = True$, and examine some values and their corresponding SDRs:

```
1  →  [1 1 0 0 0 0 0 0 0 1]
2  →  [1 1 1 0 0 0 0 0 0 0]
3  →  [0 1 1 1 0 0 0 0 0 0]
9  →  [0 0 0 0 0 0 0 1 1 1]
```

As the example illustrate, the active bits for the number **1** and the number **9** share a wrapping bit.

5.3.2.2 Categorical Data Types

NuPIC offers a categorical encoder called Category encoder which, as the name suggests, encode categorical data inputs. As previously mentioned, a good encoder should capture

the semantic meanings for the input data. This could be the case for some categorical data, as will be seen later in this Section, but usually the categorical data is discrete and unrelated. The Category encoder works well for unrelated data types and its interface has the following parameters: `w` and `categoryList`. It does not require the total number of bits for the SDRs because that can be derived from the number of the categories list and the width of the active bits (w). A concrete example could shed some light on how this encoder works. Let us take a Category encoder with the following parameters: `w=3`, `categoryList=['cat','dog']`. Here we have two categories, cat and dog. Feeding in these values to the encoder will produce the following SDRs:

```
cat   →  [0 0 0 1 1 1 0 0 0]
dog   →  [0 0 0 0 0 0 1 1 1]
other →  [1 1 1 0 0 0 0 0 0]
```

As explained earlier, no overlapping bits are there between the SDRs of `cat` and `dog` values. One interesting thing is that the first three bits are reserved to any categorical value not present in the `categoryList`.

The interface to the Category encoder suffers from an issue similar to the issue encountered with the Scalar encoder regarding the knowledge of all the possible values that could be entered to the encoder. To overcome this issue, NuPIC provides the SDR-Category encoder which allows the user to encode categorical data without knowing all the possible values that could be fed in. This is an important property to have especially if the HTM system is working in online fashion as the user cannot guarantee all possible inputs. Let us take an instance of the SDR-Category encoder with the following parameters $n = 10, w = 3$ and examine the resulting SDRs for the following values in order `['cat', 'dog', 'other']`:

```
cat   →  [1 1 0 1 0 0 0 0 0 0]
dog   →  [0 1 0 0 0 0 0 1 1 0]
other →  [0 0 0 1 0 0 0 1 0 1]
```

The SDR-Category encoder tries to assign each categorical input a unique representation. It keeps track of all the values seen so far and randomly assign unique SDRs to each value. When instantiating a new SDR-Category encoder instance with the same set of parameters and fed the same values but in a slightly different order (`['dog', 'cat', 'other']`), the following SDRs will be produced:

```
dog   →  [1 1 0 1 0 0 0 0 0 0]
cat   →  [0 1 0 0 0 0 0 1 1 0]
other →  [0 0 0 1 0 0 0 1 0 1]
```

As illustrated by the last two examples, the SDR-Category encoder assign unique SDRs and tries to neglect and minimise any overlapping active bits between them.

One of the common data types that can be found in many dataset are date and time fields. This type of data can come in many flavours and variations. For instance, there can be a `days` field in a dataset which indicates the day of the week in which a data point is recorded. For such field, the whole column can be treated as a categorical data and the Category encoder can be used to do the job. But sometimes the HTM system could use more information from the `days` field, such as knowing which time of the day the data was recorded. If the application at hand requires this level of understanding, NuPIC provides a Date encoder which has many parameters that can allow the HTM system to gain more insight and information out of the data.

The Date encoder is actually an abstraction over the normal Scalar encoder with the `periodic` flag turned on and other useful parameters that allows the user to customise the encoder according to the needs of the application. Here are all the parameters that can be specified for the Date encoder:

- `timeOfday`: Which specifies the time during a day in hours (midnight = 0).
- `dayOfWeek`: Which specifies which day it is (Monday = 0).
- `weekend`: A boolean value which indicate whether the date is a weekend or not.
- `holiday`: A boolean value which indicate whether the date is a holiday or not.
- `season`: Which indicates the season in the year whether is summer, winter, etc.

The above parameters are not mutually-exclusive and can be used in conjunction with each other. Depending on the needed properties of the input, the user can specify the applicable parameter. Let us take a couple of date and time entries and pass them to a Date encoder with the following parameters `timeOfday=3`:

```
2016-02-01 10:30 Am  →  [0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0]
2016-02-01 11:30 Am  →  [0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0]
2016-02-01 12:30 Pm  →  [0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0]
2016-02-01 11:30 Pm  →  [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
```

Every hour the resulting SDRs will shift and slide the active bits according to the time of the day. As mentioned earlier, the Date encoder uses the normal Scalar encoder with the `periodic` command set to `True` as indicated by the last entry which shows how the active bits are wrapping around as we get closer to the midnight time.

When specifying more than one parameter to the Date encoder, the encoder will generate separate SDRs for each parameter and then combine and concatenate all the SDRs into one unified representation. Let us take a Date encoder with the following parameters `timeOfday=3, weekend=3`:

```
2016-02-01 10:30 Am  →  [111000000000111000000000]
2016-02-06 10:30 Am  →  [000111000000111000000000]
```

The first date entry is Monday, February 1st which is a weekday. The second date entry is Saturday which is a weekend, the two entries have the same time. It is possible to see that the encoder allocated the first six bits to express this fact about the date. Then, the encoder concatenated the weekend/weekday representation with the time representation and unified them into one big SDR. The same procedure is followed for the rest of the available parameters.

5.3.2.3 Specialised Encoders

Having a good encoder is an important and essential requirement for the rest of any HTM system to function well. The encoders presented so far are good for most cases and applications. But there are some data types where these generic encoders will not capture the semantic qualities of the input. It is generally advised to develop a custom encoder that adheres to the required properties mentioned earlier in this Chapter, for these types of problems that cannot work well with the generic encoders.

An example of these specialised encoders is the Geo-spatial Coordinate encoder which, as the name suggests, allows the user to encode Global Positioning System (GPS) data. It also can model the speed of the object in conjunction with the location. The Geo-spatial Coordinate encoder is actually a special case of the more general Coordinate encoder which allows the user to model any number of coordinates in their HTM systems Purdy (2016).

Natural language can be encoded into SDRs and a prominent work in this field is what De Sousa Webber (2015) have been doing. Discussing the implementation of their work is outside of the scope of this project, but it is worth noting how they adhered to the required properties of a good encoder as mentioned earlier. In Figure 5.3, it can be seen that three SDRs rendered as 2D arrays. The blue and red dots represent active bits. The first SDR is for the word 'Apple', the second SDR is for the word 'Fruit', and the last SDR is for the word 'Computer'. To illustrate how this encoding technique captures a deeper semantic meaning for the respective words, the 'Fruit' SDR can be subtracted

from the 'Apple' SDR and the resulting SDR is the 'Computer" SDR. This procedure takes the *fruitiness* out of the word 'Apple', which will leave us with something referred to as an 'Apple' but it is not a fruit. Meaning Apple, the computing company. Also, the SDR of the word 'Computer' is very similar to other words shown in the figure, like Macintosh, Linux, Operating system, etc.

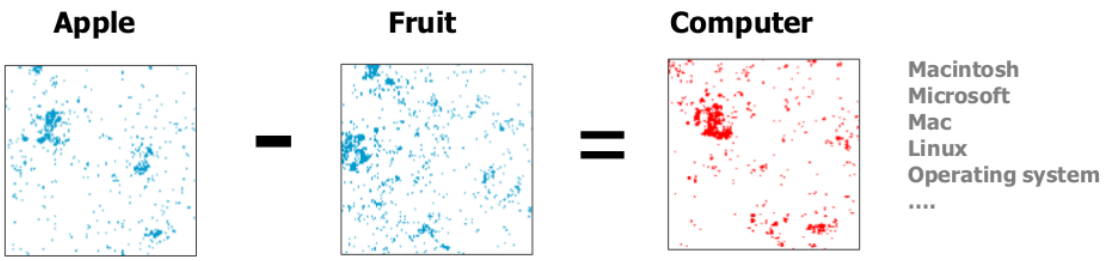


FIGURE 5.3: Using SDRs to encode natural language (Jeff Hawkins, 2014).

5.4 Smart Home Dataset and NuPIC Encoders

All the previously mentioned encoding techniques are suitable for a single dimension (or a single column) of input data to the HTM system. The advised way of encoding multiple values is to concatenate each dimension SDR and merge them into one SDR that gets fed to the HTM system sequentially. However, having too many dimensions could make it difficult for any machine learning model to learn from the data and will lead to what is known as 'The curse of dimensionality' (Bellman, 2015). To overcome this issue, it is possible to increase the training samples which can be a costly option and might not be practical depending on the application at hand. A more practical option is to try to reduce the number of dimensions by utilising techniques such as Principal component analysis (PCA) or using feature selection techniques that will put more weight or eliminate features (columns) with little or no contributions to the accuracy of the model.

As shown previously in Chapter 4 in Table 4.4 the dataset has twenty-nine binary sensors that are fed to the HTM system. The question now becomes, how can we represent or encode this multi-dimensional input? In the HTM literature, the usual solution to this problem is to concatenate the output of several encoders into one SDR. But the question still remains, what type of encoders can be used to encode each column? The scalar encoders can be used to represent the binary state of each sensor and also the categorical encoders. The important thing here is to capture the required properties for good encoders mentioned earlier in Section 5.3.1.

Let us explore the standard NuPIC encoders with the smart home dataset. Each record in the dataset represents all the states of the sensors in the smart home, captured every second. Let us assume having a Scalar encoder for each column in the dataset with these parameters `n=6, w=3, minVal=0, maxVal=1`, figure 5.4 shows an illustration of this process. The author feeds each record of the dataset to the twenty-nine scalar encoders and their output, the small 6-bits long SDRs are concatenated into one large 174-bits long SDR.

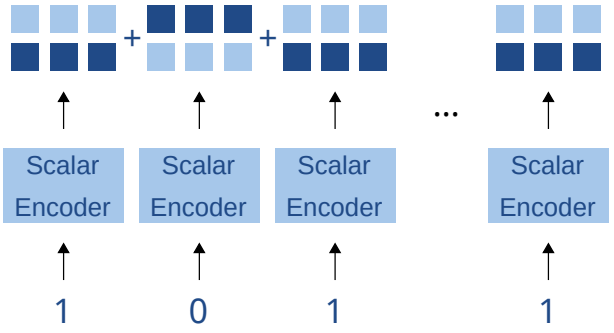


FIGURE 5.4: Using multiple scalar encoders to encode the smart home dataset.

That was one way to encode the dataset. Another way is to use category encoders instead of scalar encoders. Using the parameters `w=3, categoryList=[0, 1]`, the results are illustrated in figure 5.5. The results are very close to the result of using scalar encoders with the exception of adding 3-bits to each column which will represent any input that is not in the `categoryList`. This is how the category encoder operates (see section 5.3.2.2). The resulting SDR will be 261-bit long SDR.

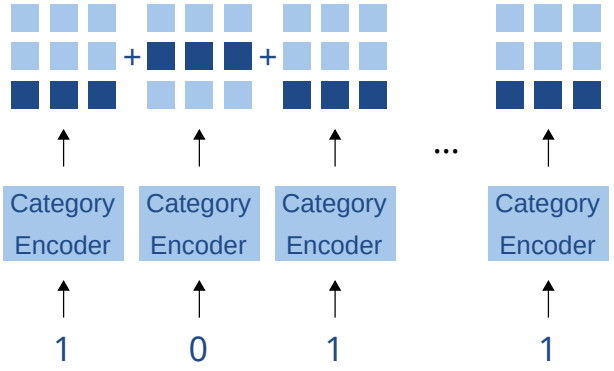


FIGURE 5.5: Using multiple category encoders to encode the smart home dataset.

To test the performance of detecting anomalies of the previous encoders, the author will evaluate their performance using an anomaly scoring metric based on NAB (see Section 6.2.3.1). The author will be using forty-two datasets and identical HTM model with identical parameters. Note that our test methodology using NAB is a bit harsh because

there is one anomaly per dataset and taking into account the false positives. Also, due to the nature of anomalies, the probability to get good scores is low.

5.4.1 Scalar Encoder

The Scalar encoder parameters that were used in these experiments are: `n=6, w=3, minVal=0, maxVal=1`. The following are several records that were fed to the encoders and the resulting SDR:

Record [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0]

Record [1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0]

As shown in the previous examples, a 29-bit long record will produce 174-bit long SDR. Every digit in the record is mapped to six bits of the resulting SDR. Any column with the value `0` will get this encoding `[1 1 1 0 0 0]` and this encoding `[0 0 0 1 1 1]` for any `1` value.

54 bits have been allocated for the timestamp column and the Date encoder was used and concatenated with the above SDR. The HTM model parameters used are:

```
SpatialPooler(
        inputDimensions=(300,),
        columnDimensions=(300,),
        synPermConnected=0.2,
        synPermActiveInc=0.003,
        synPermInactiveDec=0.0005,
        globalInhibition=True,
```

```
        numActiveColumnsPerInhArea=40,
        maxBoost=1.0,
        potentialPct=0.8,
        seed=1956)

TemporalMemory(
        columnDimensions = (300,),
        cellsPerColumn= 32,
        initialPermanence=0.21,
        minThreshold=10,
        maxNewSynapseCount=20,
        permanenceIncrement=0.1,
        permanenceDecrement=0.1,
        activationThreshold=13,
        maxSegmentsPerCell=128,
        maxSynapsesPerSegment=32,
        seed=1960)
```

LISTING 5.1: The SP and TM parameters

5.4.2 Category Encoder

Using the Category encoder is not much different from using the Scalar encoder. The only difference is that instead of having 6-bit long SDRs per column, it produces 9-bit long SDRs because it uses an additional w-bits long for the 'unknown' class as explained in Section 5.3.2.2. The following parameters `w=3, categoryList[0, 1]` were used and using the same parameters for the HTM model that were used in the previous experiment. Here are two samples that show the final SDRs:

Record [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0]

Record [1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0]

The columns with the value `0` will get this encoding `[0 0 0 1 1 1 0 0 0]`. The columns with value `1` will be encoded as `[0 0 0 0 0 0 1 1 1]`.

5.4.3 SDR-Category Encoder

In this experiment, the SDR-Category encoder was used with these parameters $n = 6, w = 3$ and used the same HTM model used in the previous two experiments.

Record [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1
0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1]

Record [1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0]

⇓

Final SDR [0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1
1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1
0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1]

As shown above, the SDR-Category encoder assigned any column with the value 0 this encoding `[1 0 0 1 0 1]` and any column with the value 1 this encoding `[0 0 0 1 1 1]`.

5.4.4 Results

Table 5.1 shows the results of the previous three experiments. The reported scores is the average score that each encoder achieved on the forty-two datasets. The low scores obtained could be attributed to the SP and the TM parameters that were used (see Listing 5.1). The parameters that are recommended[1] for doing anomaly detection on datasets that have two columns were used, a timestamp and a scalar value. These parameters are not optimised for our dataset and in the following Chapter these parameters will be explored in more details. The only change the author made is changing the input space and the columns dimension from 2048 bits to 300 just to make the experiments easier to explore and explain. The author used the same SP and TM parameters across all experiments so that the only factor affecting the results is the encoder.

TABLE 5.1: The results of using the Standard NuPIC encoders.

Encoder	**Avg Score**
Scalar encoder	14.98%
Category encoder	10.0%
SDR-Category encoder	26.46%

5.4.5 Analysis

From the previous tests, it can be seen that the encoders are struggling to produce good results. The best results are obtained from the SDR-Category encoder in Table 5.1. To analyse the issues with these encoders, there is a need to investigate their qualities against the recommended good qualities for any encoder (see Section 5.3.1).

All the tested standard NuPIC encoders ensure that a given input will get the same representation. Thus, preserving the deterministic quality of a good encoder. Also, all the encoders tested do have fixed number of bits at all times and have fixed sparsity. The issue with the previous encoders is in the quality of a good encoder, i.e. capturing semantics.

Capturing the semantics between two inputs, or in other words, any two *similar* inputs should have similar SDRs, is the problem with the previous encoding approaches. To understand this issue a bit more clearly, if we refer back to figure 5.2 where there are four SDRs that encode the scalar values $1, 2, 3, 100$. Given that the minimum and maximum values are known, which sets the context for deciding if two inputs are semantically

[1] https://github.com/numenta/nupic/blob/master/src/nupic/frameworks/opf/common_models/anomaly_params_random_encoder/best_single_metric_anomaly_params_tm_cpp.json

similar or not, it can be said that the number 1 and 2 are semantically similar. Thus, they should have similar SDRs with high number of overlapping bits. As shown in figure 5.2, it can be seen that the number 1 and 2 share the most number of overlapping bits which is two bits. However, this is not always the case. For instance, when having the minimum value set to be 1 and the maximum value set to be 2, the two numbers should not share any overlapping bits.

The concept of semantically similar inputs is domain specific. It is well-defined in the case of scalar numbers when we know which contexts these numbers are in. Another example that touches on this idea is what was shown in section 5.3.2.3 and the work of (De Sousa Webber, 2015) to encode the semantic meanings for English words.

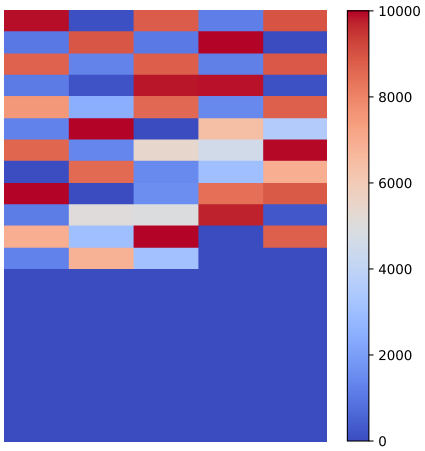
In our smart home case, what makes two inputs similar? For example, considering the following records:

Record		TV	bed	couch	fridge	main door
A	$\rightarrow$	0	1	0	0	0
B	$\rightarrow$	0	1	0	0	0

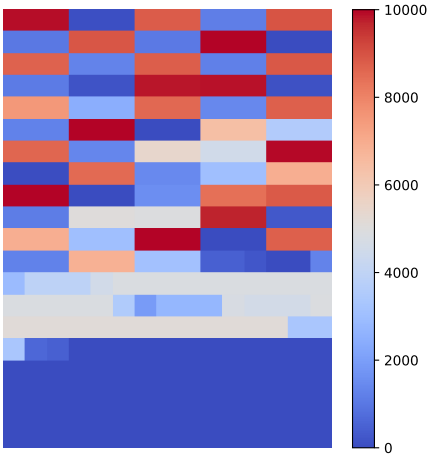
It can be seen that the bed sensor is active in record **A** and **B**. Which means, the inhabitant is laying on the bed. From the HTM system point of view, the record **A** and **B** are identical, and they will get the same SDR. However, it cannot be known if the inhabitant is sleeping or taking a quick nap or if he/she is awake and lying on the bed. There is not enough information to distinguish between 'sleeping' and 'napping' activity, for example using our dataset. However, when adding the time dimension with the input, it can be possible to distinguish between a 'sleeping' activity and a 'napping' activity since each one has its own time frame. This issue is accounted for by concatenating the time-stamp column in the dataset with the final SDR of each record. In our tests for the NuPIC encoders, the time-stamp field was included.

The following three figures (Figures 5.6 to 5.8) show a 2D heat-map of the activities of every bit in the SDRs generated by the encoder and the SDRs generated by the SP. The SDRs are 1D vectors but converted to 2D just for illustrative purposes. The author have processed 10,000 records for all of these experiments. Looking at Figure 5.6a which shows the SDR generated using the Scalar encoder without concatenating the date column. This is just to focus on how active each bit that the encoder generates. It can be seen that the active bits are laid out in 3-bit configuration because the parameter w was set to be 3. It can also be seen that most of the bits are quite active (falling in the red spectrum). Figure 5.6b shows the SDR with the date column encoded by the Date encoder and taking the bottom portion of the whole SDR. It can be seen that the

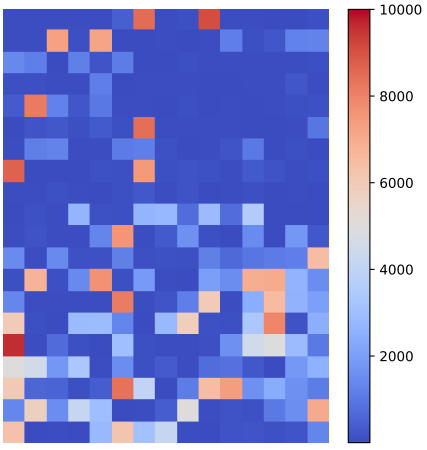
bits activities are higher in the top portion than the bit activities for the date encoding. Figure 5.6c shows the SDR that the SP is outputting. This SDR can be thought of as a top-down view on the active columns and their activities. The columns activities are scattered and most of it is happening in the bottom portion which corresponds to the date column. This is an indication that the bottom portion is actually more specialised in recognising the date column because of the topological configurations of the input space with the SP. As explained in Section 3.3, when overlaying the input space and the generated columns from the SP, every bit in the input space is more likely to form connections with the neighbouring columns in a certain radius. Therefore, it can be concluded that the date portion of the input space is more connected than the sensors portion.



(A) Scalar encoder SDR without date.



(B) Scalar encoder SDR with date.

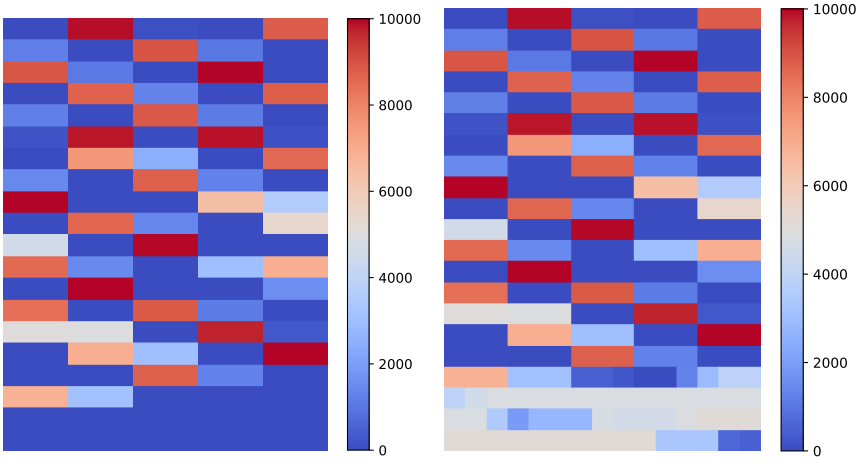


(C) The SP active columns.

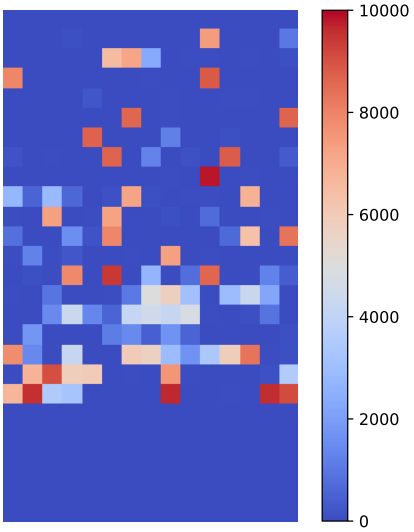
FIGURE 5.6: The activities of each bit of the SDRs after reading 10,000 records using the Scalar encoder.

The Category encoder produced similar results as shown in Figure 5.7 which was expected since the Category encoder is very similar to the Scalar encoder with our parameters.

The SDR-Category encoder also did not produce much different results. The activities of the bits are more scattered as shown in Figure 5.8a which was the expected behaviour as explained in Section 5.3.2.2. The impact on the active columns is also very similar to what was produced from using the previous two encoders.

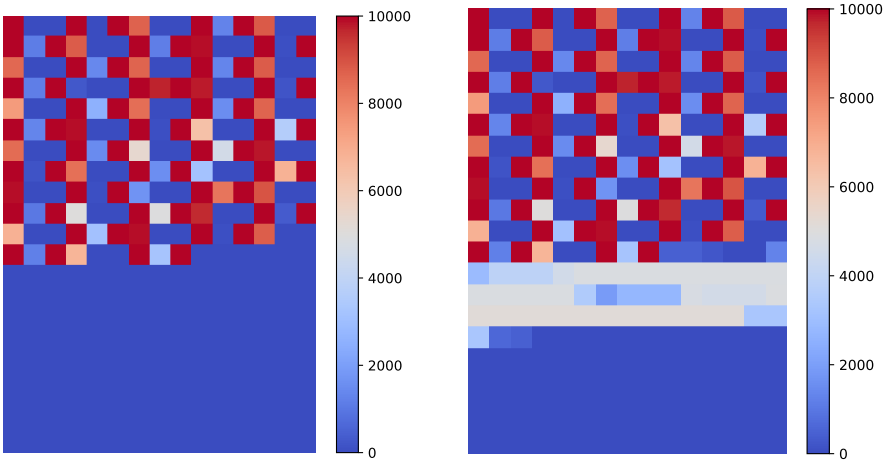


(A) Category encoder SDR without date. (B) Category encoder SDR with date.

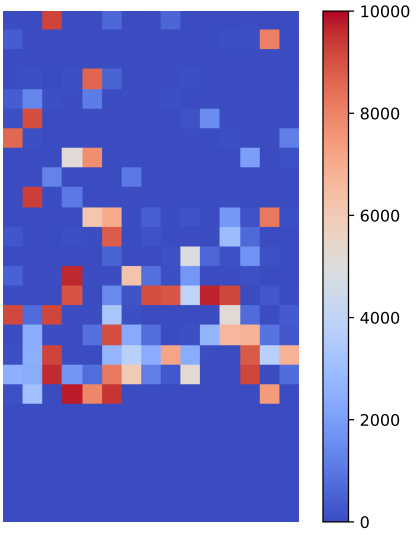


(C) The SP active columns.

FIGURE 5.7: The activities of each bit of the SDRs after reading 10,000 records using the Category encoder.



(A) SDR-Category encoder SDR without date. (B) SDR-Category encoder SDR with date.

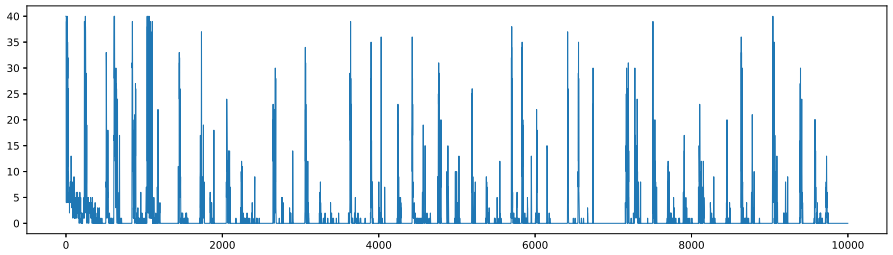


(C) The SP active columns.

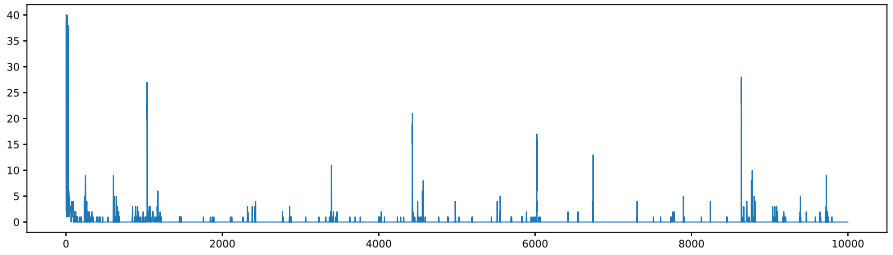
FIGURE 5.8: The activities of each bit of the SDRs after reading 10,000 records using the SDR-Category encoder.

There is another indicator that could give us some insights on what is happening in the previous three experiments. That indicator is the number of the bursting columns generated by the TM. As explained in Section 3.3, a column is said to be a bursting column when all the cells (or the bits) are activated after their SDR is processed by the TM. The columns burst when the TM cannot recognise a sequence. Figure 5.9 shows the activities of the bursting columns while reading the 10,000 records from the dataset. the total number of bursting columns the Scalar encoder generated is 15,856 columns. The Category encoder generated 2,262 columns. The SDR-Category encoder generated 4,741 columns. It is worth noting that having less bursting columns is not always an indication that the system is performing well. It means the system learned something,

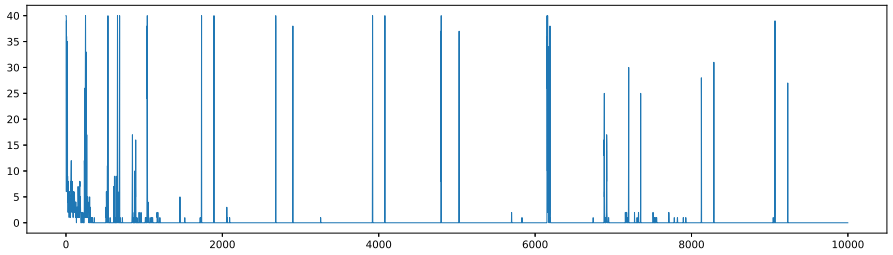
but we do not know if it is the right thing or not. In other words, it shows that the system is not surprised by the input. Of course, we do not need a system that it is too relaxed nor a system that is too sensitive, the biasvariance trade-off comes to play in this situation.



(A) Scalar encoder bursting columns.



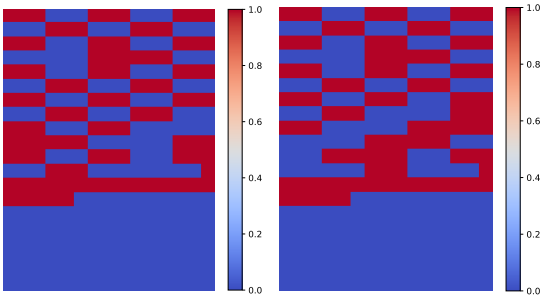
(B) Category encoder bursting columns.



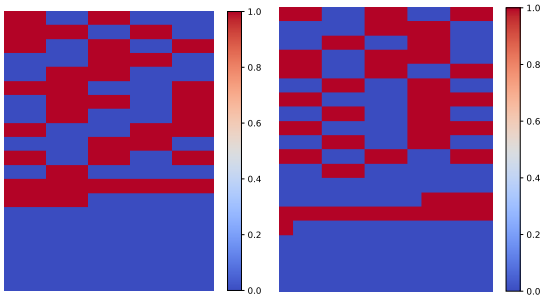
(C) SDR-Category encoder bursting columns.

FIGURE 5.9: The bursting columns activity after reading 10,000 records.

To get more insights let us investigate a sequence of the inhabitant's activities and the representations that the encoders produced for each activity. For example, assuming the following sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$, the following figures (Figures 5.10 to 5.12) show the sequence and the SDRs using the three encoders used in our experiment.

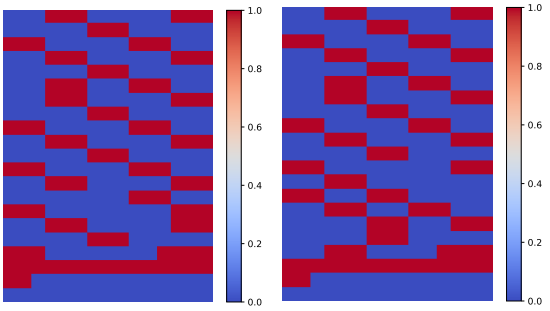


(A) Sleep SDR. (B) Personal SDR.

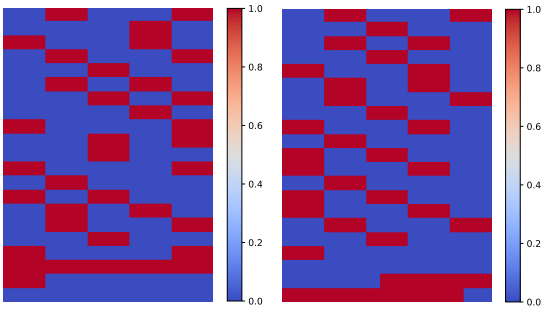


(C) Eat SDR. (D) Work SDR.

FIGURE 5.10: The sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$ SDRs produced by the Scalar encoder.

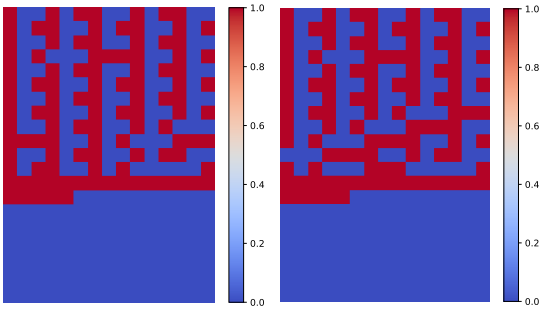


(A) Sleep SDR. (B) Personal SDR.

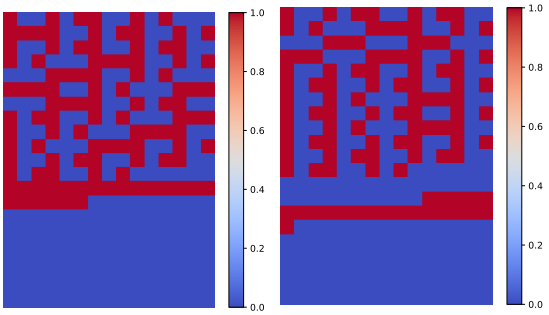


(C) Eat SDR. (D) Work SDR.

FIGURE 5.11: The sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$ SDRs produced by the Category encoder.



(A) Sleep SDR. (B) Personal SDR.



(C) Eat SDR. (D) Work SDR.

FIGURE 5.12: The sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$ SDRs produced by the SDR-Category encoder.

From the previous figures, it can be seen that the main problem with standard NuPIC encoders. For each encoder, the SDRs generated for different activities (e.g. $sleep, eat, etc.$) look very similar. This representation makes it hard for the HTM system to recognise the activity spatially and hence, making it harder to recognise the overall sequence over time. In the following Section, we present our novel encoder that overcomes the problems presented in this Section and their results.

5.5 The HI-SDR Encoder

The author developed a novel encoder, the Hash-Indexed Sparse Distributed Representation (HI-SDR) encoder, that adheres to the required properties of good encoders mentioned in Section 5.3.1. Our encoder is deterministic. Given two identical inputs, it will produce the same SDR for both inputs. The sparsity and the number of bits are fixed thus meeting the first three requirements for good encoders. The problematic property for the encoders explored in the previous Section is the fourth property, capturing semantics. The standard NuPIC encoders produced very similar SDRs for very different inputs. Thus, confusing the HTM system and not giving it enough information to distinguish and recognise the inputs.

To meet the requirement for the fourth property, there is a need for a way to encode every record uniquely and maintain the determinism and sparsity of the generated SDR. This research proposes a solution using a hash function and passing it the input records to produce a digest. The author used the produced digest to position w-bits in the resulting SDR uniquely. When encountering the same record again, the hash function will produce the same digest. Thus, maintaining the determinism and creating unique SDRs for every input configuration.

Figure 5.13 shows how the HI-SDR encoder works at a high level. It takes a whole dataset record (e.g. $[0, 1, 0, 0, \ldots, 0]$) and pass it to a hashing function that generates a hash digest. The digest is used then to place the active bits in the SDR. By taking every digit of the digest and using it as the index of the active bit placement, a result similar to what is shown in Figure 5.13 is obtained. There are several parameters for the HI-SDR encoder explained in detail in Section 5.5.2.

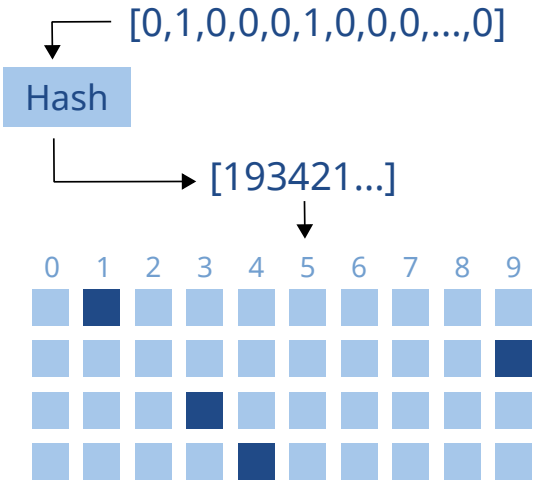


FIGURE 5.13: The HI-SDR encoder.

The author tested and experimented with different hashing functions. The requirements, in this research, were speed and how random is the produced digest. For the speed requirement, the author excluded all cryptography hashing functions as they are slow by design to make it hard to crack. The author experimented with several non-cryptographical hashing function such as the Python implementation of CRC32[2], and Adler-32[3]. The author decided to use the xxHash (Collet, 2015) function due to its speed and the randomness of the generated digests.

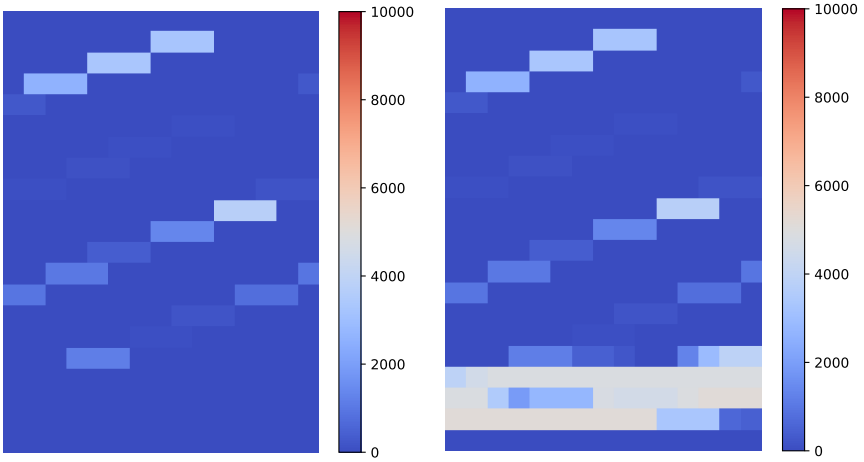
[2]https://docs.python.org/2/library/zlib.html#zlib.crc32
[3]https://docs.python.org/2/library/zlib.html#zlib.adler32

5.5.1 HI-SDR Encoder Results

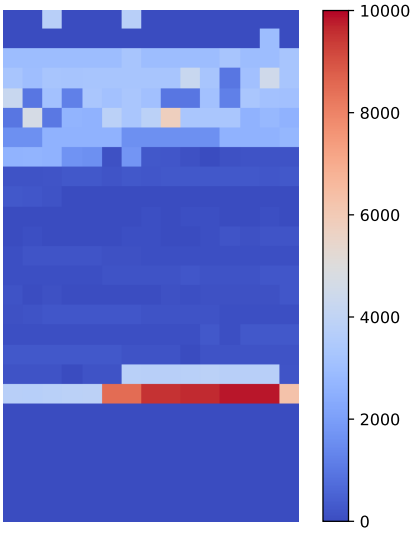
The HI-SDR encoder scored 33.28% on the same forty-two datasets used to test the standard NuPIC encoders. The best score for the standard NuPIC encoders was by SDR-Category encoder which scored 26.46% (see Section 5.4.4). The parameters for HI-SDR encoder were $n = 300, w = 3, p = 2$. Again, an unoptimised parameters for the SP and the TM were used which affects the results. Later on in this Section, the results of using an optimised parameters will be presented to show the potential of the proposed encoder. The author used the same (unoptimised) parameters for all four encoders just to take the SP and the TM parameters out of the equation and make the encoder solely the factor affecting the score.

It can be seen that the drastic difference in the heat-map shown in Figure 5.14 of the proposed encoder compared to the standard NuPIC encoders. In Figure 5.14a, which does not include the date column, it can be seen how active the input bits are. The activity is more spread across the SDR space. In Figure 5.14b, which does include the date column, it can be seen that most of the activity is happening in the bottom portion of the SDR. Unlike the standard encoders, the upper portion is less active in the proposed encoder. This is a more faithful representation of what is happening in the dataset. The time-stamp column is presented in every record and it is cyclical and wraps around. Therefore, it covers the same area in the input space. Compare that to the standard encoders SDRs which have the upper portion much more active than the date portion.

Figure 5.14c shows the SP active columns. It can be seen how balanced is the activity between the upper portion (which is corresponding to the sensors input), and the lower portion. This means that the SP is recognising and learning from both portions, giving a better picture for the HTM system.



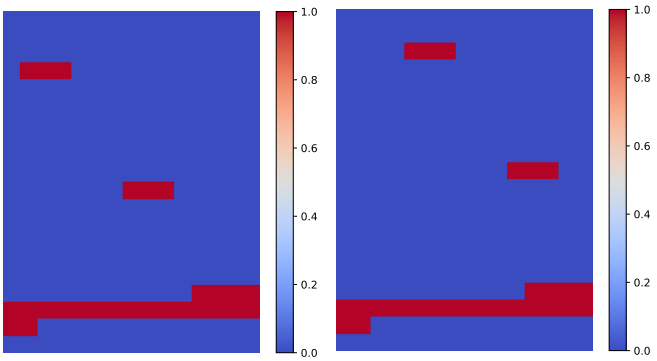
(A) HI-SDR output without date. (B) HI-SDR output with date.



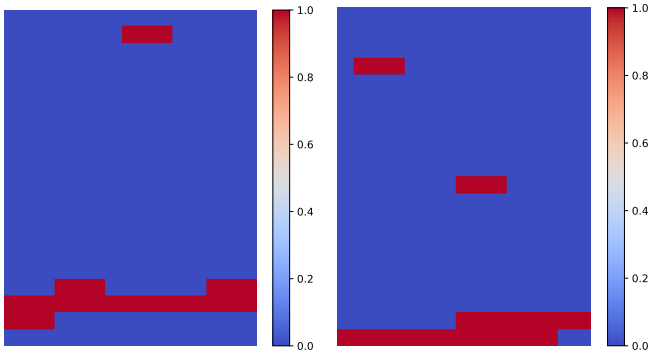
(C) The SP active columns.

FIGURE 5.14: The activities of each bit of the SDRs after reading 10,000 records using the proposed HI-SDR encoder.

Figure 5.15 shows the same sequence that was explored in the previous Section ($sleep \rightarrow personal \rightarrow eat \rightarrow work$) and the resulting SDR for every activity performed by the inhabitant. It can be seen how the proposed encoder uniquely assigns a representation for every activity. The proposed encoder uses different places to position the active bits across the input space. This helps the SP to learn much faster and distinguish the inputs more clearly compared to the standard encoders.



(A) Sleep SDR. (B) Personal SDR.



(C) Eat SDR. (D) Work SDR.

FIGURE 5.15: The sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$ SDRs produced by the HI-SDR encoder.

The proposed encoder is capable of uniquely representing any input regardless of the input density. A record that contains no active bits (e.g. $[0, 0, 0, 0, ..., 0]$) and a record that is fully dense (e.g. $[1, 1, 1, 1, ..., 1]$), both will get a unique SDR with fixed sparsity and fixed active w-bits. Thus, fulfilling the required properties of a good encoder. This is important if we want the encoder to work with any number of sensors in an unsupervised fashion.

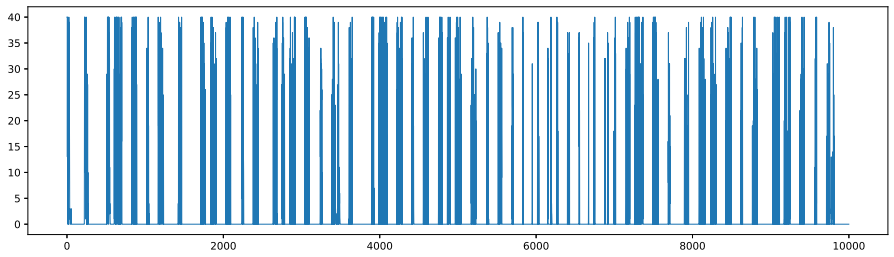


FIGURE 5.16: The bursting columns activity produced by the HI-SDR encoder.

Figure 5.16 takes a look at the bursting columns while processing the 10,000 records of the dataset. It can be seen that the columns are bursting quite often. In fact, the total number of the bursting column throughout the experiment was 28,045 columns. The highest number of bursting columns was the Scalar encoder with 15,856 bursting columns. This in an indication that the parameters of the SP and the TM are not tuned for this dataset. However, even with this misconfigured parameters, the HI-SDR encoder outperformed the standard encoders.

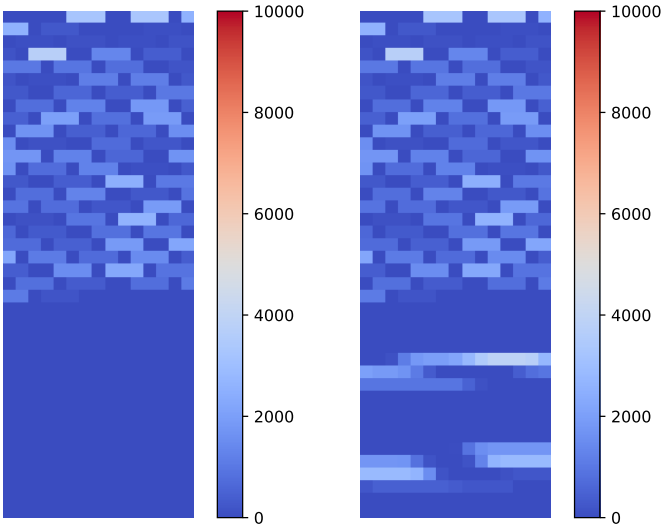
```
SpatialPooler(
        inputDimensions=(600,),
        columnDimensions=(600,),
        synPermConnected=0.1,
        synPermActiveInc=0.05,
        synPermInactiveDec=0.00005,
        globalInhibition=True,
        numActiveColumnsPerInhArea=44,
        maxBoost=20.0,
        potentialPct=0.8)

TemporalMemory(
        columnDimensions = (600,),
        cellsPerColumn= 10,
        initialPermanence=0.21,
        connectedPermanence=0.1,
        minThreshold=4,
        maxNewSynapseCount=20,
        permanenceIncrement=0.5,
        permanenceDecrement=0.001,
        activationThreshold=9)
```

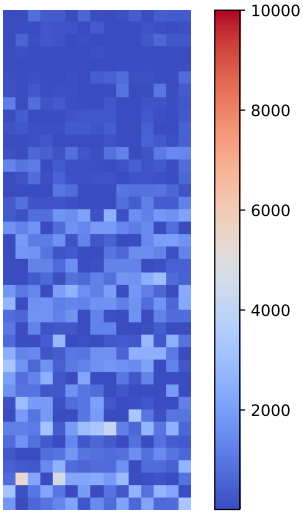
LISTING 5.2: The optimised SP and TM parameters

Just to see the potential of the proposed encoder, the author ran another experiment using the same forty-two datasets with an optimised parameters for both the SP and the Temporal Memory. The HI-SDR encoder scored 76.59% using the parameters shown in Listing 5.2. The parameters for the HI-SDR encoder were $n = 600, w = 3, p = 8$.

Figure 5.17 shows the SDRs with the optimised parameters and the active columns generated by the SP. Figure 5.18 shows the SDRs for the sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$, as done before. The author notice that the SDRs are more active than the previous experiment and that is due to the number of partitions (e.g. the p parameter of the proposed encoder). The details of the HI-SDR parameters and the algorithm are presented in Section 5.5.2.

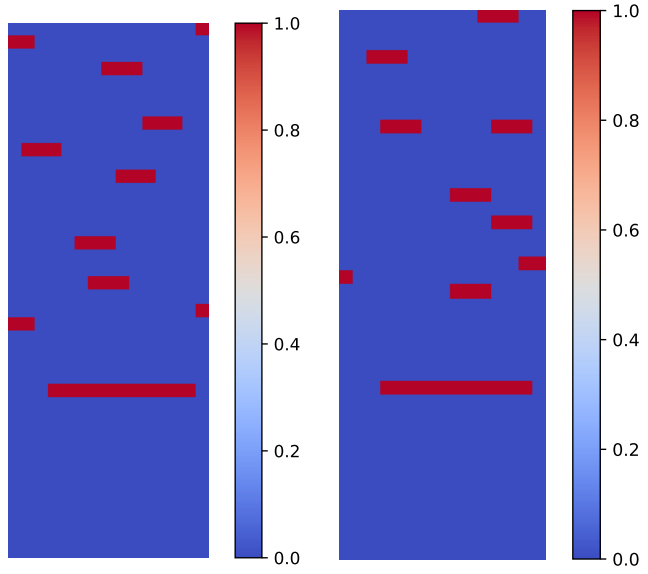


(A) HI-SDR output without date. (B) HI-SDR output with date.

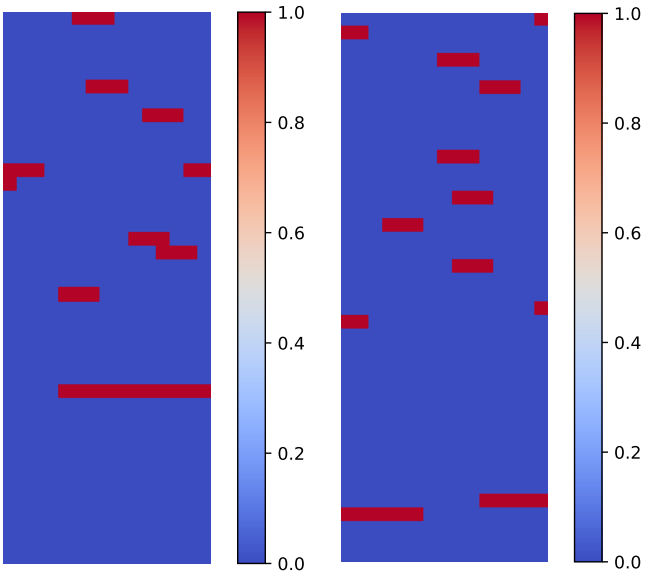


(C) The active columns.

FIGURE 5.17: The HI-SDR results with the optimised parameters for the SP and the TM.



(A) Sleep SDR. (B) Personal SDR.



(C) Eat SDR. (D) Work SDR.

FIGURE 5.18: The sequence $sleep \rightarrow personal \rightarrow eat \rightarrow work$ SDRs by the HI-SDR encoder with the optimised parameters.

The total number of the bursting columns with the optimised parameters is 4,531. This shows a large difference than the old number with the unoptimised parameters which was 28,045 columns. Moreover, Figure 5.19 shows the behaviour of the bursting columns, while processing the 10,000 records. It can be seen that this gradual decrease in the number of bursting columns which is an indicator that the HTM system is actually learning the pattern that the inhabitant is doing throughout the day.

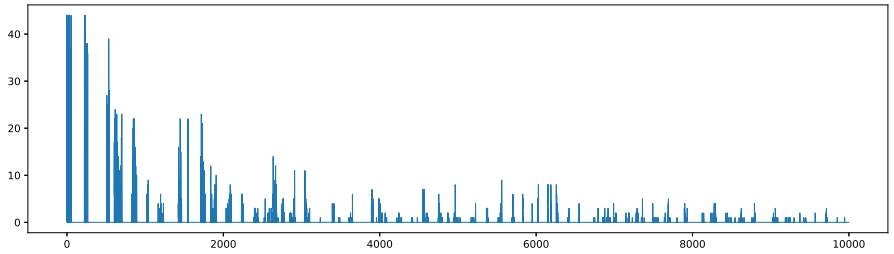


FIGURE 5.19: The bursting columns activity by produced the HI-SDR encoder with the optimised parameters.

5.5.2 The Algorithm

The algorithm can be divided into two parts, the hashing part and the SDR construction part.

For the first part, the array of the sensors readings were fed to the hashing function to obtain the hash digest. The author has tested several non-encryption hashing functions and the results of the `xxHash` function produced the best results on our datasets.

For the second part, which is the construction of the SDR based on the hash digest. Algorithm 1 presents the pseudo-code of the actual implementation of construction algorithm. The algorithm receives four parameters, namely:

- **hash**: which is the hash digest.
- **n**: the total number of bits of the SDR.
- **w**: the number of active bits in the SDR.
- **p**: the number of the partitions.

Algorithm 1 SDR construction algorithm.

```
1: procedure CONSTRUCT-SDR(hash, n, w, p)
2:     SDR ← [0] * n                          ▷ SDR is n length zero array
3:     skip ← INT(n/partitions)
4:     hashDigits ← STR(hash)                 ▷ Converts the hash digits to string
5:
6:     if w > (skip/10) then                  ▷ Divide by 10 because there are 10 digits
7:         Raise ValueError
8:     end if
9:
10:    for d in hashDigits do
11:        i ← INDEX(d)
12:        if i == p then
13:            break
14:        end if
15:
16:        ri ← INT(d) + 1                    ▷ Calculating the relative index position
17:        pct ← FLOAT(ri)/10
18:        ri ← ROUND(skip * pct) − 1
19:
20:        for j in RANGE(w) do
21:            diff ← (w − 1)
22:            SDR[ri + (i * skip) + j − diff] = 1
23:        end for
24:
25:    end for
26:    return SDR
27: end procedure
```

5.6 Summary

In this Chapter, a formal definition of the SDRs and the mathematical notations of this concept were presented. The SDRs properties and the operations that can be performed with them were explored. The required properties for the encoders to produce good SDRs were explained. The most challenging property was capturing the semantic meaning where any two similar inputs should result in similar SDRs. The other required

properties were having a deterministic encoder, having a low density of active bits, and having a fixed number of total bits generated by the encoder.

Standard NuPIC encoders for different numerical, categorical and specialised data types were explored. The smart home datasets and the problems with using standard NuPIC encoders to represent the datasets records were presented. Then, several experiments were conducted and how standard encoders do actually encode the datasets' records and the effects for these encoding on the SP active columns and on the TM bursting columns. Guided by the required properties for good encoding, the author proposed and developed a novel encoder and compared its results with the standard ones. The Chapter concludes with the algorithm for the hash encoder.

The experiments shown in this Chapter were conducted to highlight the issues with the standard encoders are very brief and lack the needed depth and analysis. The next Chapter will evaluate the standard encoders and the novel encoder thoroughly and will present the adopted evaluation methodology.

Chapter 6

Test and Evaluation

6.1 Introduction

In this Chapter, the evaluation methodology adopted in this study is presented by showing the virtual smart home plan designed using OpenSHS and the chosen context to be simulated by the participants. Following that, the datasets generated by the participants and the type of anomalies are presented. A detailed explanation of how the experiment is designed by showing the anomaly scoring methodology and the different experiment parameters are presented. The results of executing several unsupervised anomaly detection algorithms are reported and the Chapter concludes by a discussion of these findings.

6.2 Methodology

This Section provides details on the virtual smart home plan that was designed by the researcher for the participants to conduct their daily activities in. In Chapter 4, the concept of simulating specific context of interest can accelerate the simulation time for the participants. In this Section, a detailed discussion about the chosen contexts to simulate are presented.

The process of generating the datasets is explained in the following Section along with the procedure that was applied by the participants. The specific parameters used in OpenSHS to aggregate the datasets are also presented. The types of the anomalies in each dataset are also presented in this Section and the reasons for the author's choice of binary sensors in the virtual smart home. The Chapter concludes by a detailed explanation of the evaluation process.

6.2.1 Smart Home Design

A smart home consisting of bedroom, living room, bathroom, kitchen, and home office as shown in figure 6.1 was designed, each with several types of sensors.

FIGURE 6.1: The floor plan's design of the smart home.

The smart home is equipped with twenty-nine binary sensors as show in Table 6.1. 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 positions. An example of this type is the carpet sensors, which turn on when the participant walks over them.

It is worth noting that type of the sensors, in terms of being passive or active, has no implications on the performance of the evaluated models. This distinction is not included in the generated datasets.

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 (as discussed in Section 4.3.1.3) that were decided to be included in this dataset are: *sleep, eat, personal, work, leisure, other*. The anomalies have an additional label *anomaly*.

During the simulation, when the participants want to change their activity, they can do so by using the dialogue box shown in figure 4.4. It is worth noting that when the participants change their activity label, it does not immediately change in the dataset. The activity label changes when one of the sensors states have changed. This approach ensures a clean separation when the participants transition from one activity to another.

TABLE 6.1: All smart home sensors and dataset columns.

i	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	

6.2.1.1 Why the Sensors are Binary?

As shown in table 6.1, all the sensors in the smart home are produce output in binary. This design choice can be justified by several reasons:

- most of the sensors states can be binary,
- easier to implement in existing middlewares,
- closer to the way HTM encoders work.

Most of the smart devices states are naturally binary, including anything that can be opened and closed or anything that can detect presence or absence of some object. Of course this binary simplification can not cover all the states that a device can have, but for the purposes of anomaly detection, this is sufficient. For example, a smart TV can be in many states such as being fully on and playing a channel, it can be in a sleep mode where it uses less power, or it can be fully turned off. For detecting the daily patterns of the inhabitant it is more important to know whether the TV is on or off than knowing if the TV is off or in a sleeping mode.

If a middleware implementation wants to add an anomaly detection service using the proposed technique, all it has to do is to set thresholds either manually by the user or using more sophisticated techniques to set the thresholds for each device. Setting these thresholds will turn the output of any sensor into binary reading that can be incorporated into this work.

Working with binary data directly is closer to how the encoders in HTM systems work using SDRs. Therefore, this choice gave us more freedom to integrate all smart home outputs with the proposed HTM system.

6.2.1.2 Contexts

In this work, two contexts in the day and two contexts in the week were chosen to be simulated by the participants. Totalling four different contexts per participant. The day contexts are ‘morning’ and ‘evening’ contexts. The week contexts are ‘weekday’ and ‘weekend’ contexts.

6.2.2 The Participants and the Datasets

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

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

For each participant, the following procedures was followed:

1. The researcher guides the participant and shows him/her the virtual smart home.
2. The participant was asked to play with the virtual smart home to get familiar with it.
3. Test the participant familiarity with the virtual smart home by asking him/her to perform specific tasks.
4. The actual simulation takes place, and the participant will be asked to give their actual starting times for each context.
5. After the simulation of his/her normal behaviour, the participant is asked to perform an anomalous simulation that he/she would like the system to identify as an anomaly.
6. The participant is asked to fill the usability questionnaire.

The researcher did not impose any restrictions or limitation on the participants during the simulation phase. The participants had the freedom to perform any activity and for any length of time.

6.2.2.1 Dataset Aggregation

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

1. `days`: 30 and 60 were chosen.
2. `start-date`: 2016-02-01 was chosen.
3. `time-margin`: the values 0 minutes, 5 minutes, and 10 minutes were chosen.

The above parameters generate one month and two months worth of data. For the one-month dataset, there are 3 variants with 0, 5, and 10 time-margins. The same goes for the two-month dataset. This ensures that the generated datasets are different in the time dimension. Table 4.4 shows a sample of the final dataset. The total number of datasets is forty-two and for each one the corresponding anomalous event was injected.

TABLE 6.2: The forty-two test datasets and the corresponding number of records.

Name	Records
d1-1m-0tm	18,121
d1-1m-5tm	18,097
d1-1m-10tm	18,045
d1-2m-0tm	35,034
d1-2m-5tm	34,968
d1-2m-10tm	35,066
d2-1m-0tm	35,359
d2-1m-5tm	35,680
d2-1m-10tm	35,542
d2-2m-0tm	74,172
d2-2m-5tm	72,164
d2-2m-10tm	72,752
d3-1m-0tm	40,604
d3-1m-5tm	40,065
d3-1m-10tm	41,682
d3-2m-0tm	88,092
d3-2m-5tm	88,092
d3-2m-10tm	87,553
d4-1m-0tm	30,032
d4-1m-5tm	30,924
d4-1m-10tm	29,646
d4-2m-0tm	61,115
d4-2m-5tm	59,445
d4-2m-10tm	56,830
d5-1m-0tm	41,344
d5-1m-5tm	39,725
d5-1m-10tm	40,818
d5-2m-0tm	78,268
d5-2m-5tm	79,049
d5-2m-10tm	78,628
d6-1m-0tm	88,884
d6-1m-5tm	90,435
d6-1m-10tm	88,943
d6-2m-0tm	174,810
d6-2m-5tm	174,190
d6-2m-10tm	169,655
d7-1m-0tm	53,322
d7-1m-5tm	51,973
d7-1m-10tm	52,263
d7-2m-0tm	99,194
d7-2m-5tm	102,341
d7-2m-10tm	100,975
Total	2,743,897

The naming convention used for the dataset 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,
- `z`: is the `time-margin` value.

6.2.2.2 The Anomalies

In some contexts, the definition of an anomaly is clear and can be quantified, for example, the heart rate for a patient. A heart rate that ranges from 60 to 100 beats per minutes is considered a normal resting heart rate for an adult. However, in the context of an inhabitant's behaviour in their smart home environment, the definition of what an anomalous behaviour is? Can difficult and hard to quantify. Anomalous behaviour becomes much more subjective and varies from one inhabitant to another. It is for this reason the anomalies in the datasets were not injected, after the simulations were conducted, based on the researcher's idea of what an anomaly is.

To overcome the issues with defining what is an anomaly for an inhabitant, the researcher left this definition to the persons capable of defining these anomalies, the participants themselves.

Each participant performed an additional simulation that is intended to represent an anomaly from the point of view of the participant. All the anomalies are defined by the participants and no restrictions were imposed by the researcher. Table 6.3 shows each participant's anomaly that he/she simulated. Although there are seven anomalies in total, each anomaly is injected into six different contexts based on the user behaviour.

The anomalies in this work were defined at the level of the individuals. This decision was made because of the unnecessarily added complexity of defining the anomalies at the group/inhabitants level. As suggested by Novak *et al.* (2012), adding an RFID tag can easily identify each person behaviour in the smart home. Therefore, when the problem is solved efficiently at the individual level, this solution can be trivially extended to the group by separating each inhabitant from another by their RFID tags.

6.2.3 Experiment Design

Figure 6.2 shows the overall design of the experiment. After the preparation of the forty-two datasets, the records of each dataset are fed to a Machine Learning model.

TABLE 6.3: the anomalies defined by the participants.

participants	**anomaly definition**
participant 1	leaving the fridge door open.
participant 2	leaving the oven on for long time.
participant 3	leaving the main door open.
participant 4	leaving the fridge door open.
participant 5	leaving the bathroom light on.
participant 6	leaving tv on.
participant 7	leaving light bedroom and wardrobe open.

The model produces an anomaly score for each record it reads. No labels are fed to the Machine Learning model, as this process is completely unsupervised. The output of the model is a dataset with the records' number and the associated anomaly score. To evaluate the performance of the model, the author derived an anomaly scoring metric based on NAB (see Section 6.2.3.1) that takes the model's output and scores it based on the correct labels (ground truth). The anomaly score will be referred to as 'NAB' score throughout the thesis. The higher the score is, the better performance the model has in detecting anomalies.

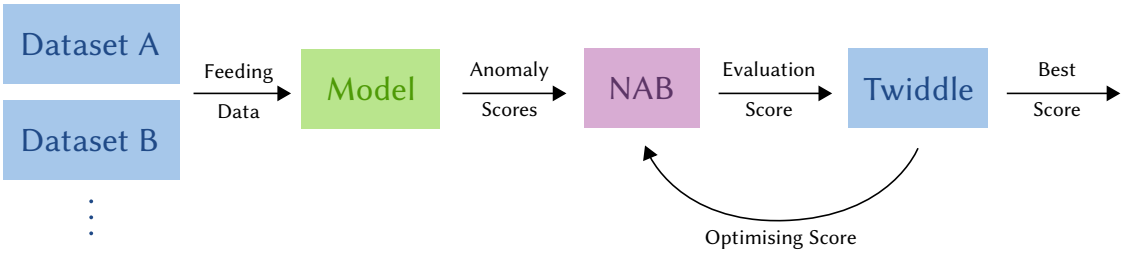


FIGURE 6.2: The experiment design.

It is worth noting that at every experiment, a set of fixed parameters for one model were used to run on all the forty-two datasets. For each of the datasets, the model is initialised and re-set as if it is the first time it sees the currently used dataset.

With the exception of DBSCAN, most of the evaluated models produce a scalar anomaly score, the question is: How can this score be interpreted? The common answer is to consult experts in the field and set a fixed threshold for which a score greater than this threshold is considered an anomaly. But there is an issue with this approach with smart home datasets. The experts are the smart home inhabitants and what is considered an anomaly for one, might not be an anomaly for another inhabitant. Therefore, having a fixed threshold for all dataset is not an optimal solution. To overcome this issue, an optimisation algorithm (for more information about Twiddle see Section 6.2.3.2)

was used. Twiddle tries many thresholds and chooses a threshold that yields the best performance. As shown in Figure 6.2, Twiddle optimises the anomaly score by iteratively trying different thresholds and reporting the best threshold as the final score.

Each tested model received all data generated from all the sensors at a fixed sampling rate (every one second). The choice for feeding all the data to the model was based on the intuition that the model should be able to capture a holistic patterns of the inhabitants. There is a natural progression of these patterns and an emerging time-dependency of these activities. If a dedicated model for each sensor was used, this time-dependency or this sequencing of activities would be lost.

The Machine Learning model produces an anomaly scalar score ranging from 0.0 to 1.0 representing how anomalous a given record is to the model. Some models has this fixed range by default and some of them does not have an upper limit of this value. For the algorithms that do not have an upper limit, the scores were normalised and limited to the range 0.0 to 1.0.

6.2.3.1 Anomaly Scoring Metric

Anomaly detection evaluation is a tricky and challenging task due to the nature of the anomalies. Traditional evaluation metrics such as recall and precision are not suitable for evaluating real time anomalies as they do not take into consideration the time aspect in the evaluation process. An early anomaly detection is better than a latter detection. This fact is not reflected and rewarded when using recall and precision evaluation techniques. Therefore, a new anomaly detection framework called Numenta Anomaly Benchmark (NAB) for real time data was proposed by (Lavin & Ahmad, 2015). The proposed evaluation method values early detection of anomalies.

NAB is an open source benchmarking framework[1] used for evaluating the performance of real time anomaly detection algorithms. NAB relies on three components to do its evaluation: *an anomaly window*, a *scoring function* and an *application profile*. The framework takes a dataset with labelled anomalies and establishes anomaly windows around every anomaly and whenever the tested algorithm detects an anomaly in that window, a positive score will be added to the algorithm's performance. If the algorithm detects multiple anomalies within the same anomaly window, only the first one will be scored. Figure 6.3 shows a sample data with anomalies marked by red dots. The red shaded regions are the anomalies" windows and the purple shaded region is a grace period where the algorithm under test is not evaluated in this period.

[1] https://github.com/numenta/nab

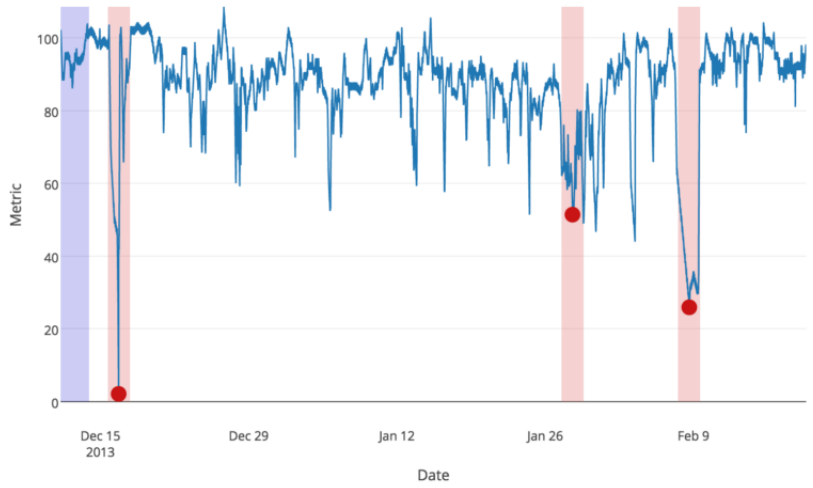


FIGURE 6.3: An example data with anomalies (Lavin & Ahmad, 2015).

The anomaly window's length is defined to be 10% of whole length of the data file divided by the number of the anomalies. This decision stems from the assumption that anomalies are rare events and the author has tested different values for the anomaly window length, and they found that 10% is a good approximation.

Figure 6.4 shows multiple anomalies (denoted by the red and green crosses) and also shows the scoring function. The first anomaly is a *false negative* that was not detected by the tested algorithm and therefore it receives a -1.0 score for that anomaly. The second anomaly was correctly detected and was detected relatively early and therefore received a +0.9999 score. The third anomaly was correctly detected but since it is within the anomaly window, its score was ignored. The fourth anomaly was not detected with high confidence by the algorithm and therefore it received a -0.8093 score.

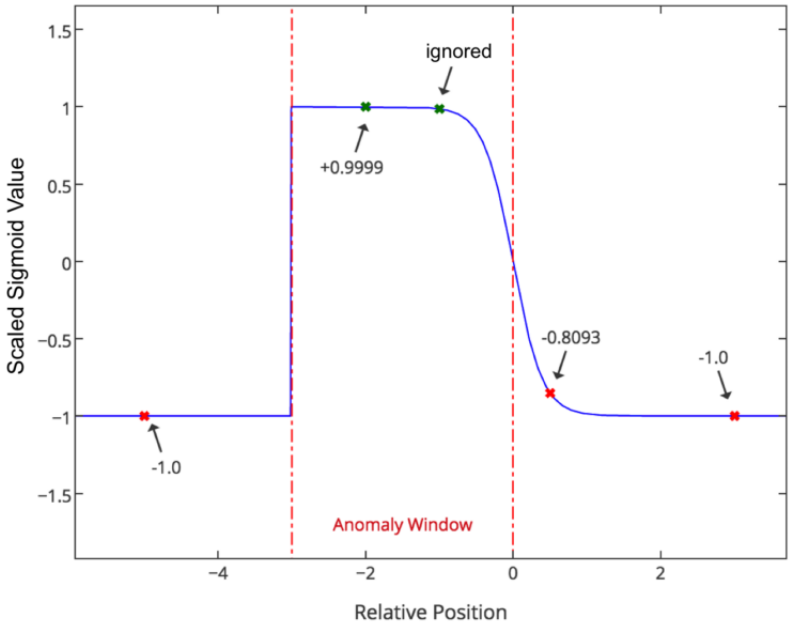


FIGURE 6.4: NAB scoring function (Lavin & Ahmad, 2015).

Some applications value *false positives* detection more than *false negatives*. NAB takes this into consideration by providing application' profiles that vary the weights of the severity of false positives and false negatives. NAB comes with three profiles:

- **Standard profile:** Which assigns equal weights for true positives, false positives and false negatives.
- **Reward low false positives profile:** Which assigns more weight and penalties for false positives.
- **Reward low false negatives profile:** Which assigns more weight and penalties for false negatives.

The NAB framework also includes datasets of real-world data and synthetic data. Every dataset defines its own anomalies. The anomalies could be spatial anomalies (point anomalies) or temporal anomalies (contextual anomalies). The datasets are coming from different domains and applications, for example, there is a dataset for CPU utilisation in Amazon Web Service (AWS). Another example is the value of a stock in the market. The diversity of the datasets adds to the generalisability test for a proposed technique. The full datasets are publicly available at Lavin & Ahmad (2015).

All of NAB datasets are time-series data and every dataset has two columns, a time stamp column and a scalar value. Some datasets contains anomalies that are known and the cause of these anomalies is identified. An example of such dataset is the `nyc-taxi`

dataset which contains a 30 minute aggregate of the number of passengers in New York City Taxis. There are five anomalies that occur which are NYC marathon, Thanksgiving, Christmas, New Years day, and in a snow storm. The other datasets that do not have a known cause of the anomalies are labelled by hand. The labelling process follows a set of steps that can be found in the project's website. The labels are not used to train any tested model as all the models are working in an unsupervised fashion. The labels are only used after running the model on the dataset to score and evaluate its performance.

6.2.3.2 Twiddle

Twiddle is a local hill-climbing algorithm that can be used to find a local optimum (Thoma, 2014). In this research it was used to find the best threshold that separates normal data points from anomalous ones.

Algorithm 2 The Twiddle algorithm.

```
1:  procedure TWIDDLE(p, dp, A, θ)
2:      bestErr ← A(p)
3:
4:      while SUM(dp) > θ do
5:          for i ← 0, LEN(p) do
6:              p[i] ← p[i] + dp[i]
7:              err ← A(p)
8:
9:              if err < bestErr then                    ▷ Found improvement
10:                 bestErr ← err
11:                 dp[i] ← dp[i] × 1.1
12:             else                                     ▷ No improvement found
13:                 p[i] ← p[i] − (2 × dp[i])            ▷ Search in the other direction
14:                 err ← A(p)
15:
16:                 if err < bestErr then
17:                     bestErr ← err
18:                     dp[i] ← dp[i] × 1.05
19:                 else                                 ▷ No improvement found
20:                     p[i] ← p[i] + dp[i]
21:                     dp[i] ← dp[i] × 0.95             ▷ Decrease step size
22:                 end if
23:             end if
24:         end for
25:     end while
26: end procedure
```

The Algorithm 2 receives four parameters, namely p, dp, A, θ. These parameters are:

- p: The initial parameters for the function that will be optimised,

- dp: The step size for the search,
- A: The error/cost function that will be optimised,
- θ: The tolerance threshold (not to be confused with the SDR's θ).

p and dp can be a single value or a vector of values for the cost function A.

6.2.3.3 Experiment Parameters

As explained earlier, every dataset has one anomaly identified by each participant according to their daily habits. The participants had total freedom to define what they would like the system to recognise as an anomaly. Therefore, every participant had different anomaly periods. To follow a similar evaluation procedure, the anomaly period was set to be at least five minutes long (300 records in the dataset as the sampling was done once every second). The modification are applied only for the labels while the sensors' readings are left intact. Figure 6.5 shows one of the datasets used in the experiment.

A probationary period that is not scored is defined for every dataset to allow the model to learn the participant's normal habits from his/her data. This probationary period is set to be the first 80% of the dataset under test. Figure 6.6 shows this period highlighted and all the anomaly metrics are set as 50% and are not scored. The scoring only happens for the last 20% of the dataset.

Figure 6.7 shows a zoomed view on part of the last 20% of the dataset. The red-highlighted section is what the participant defined to be the anomaly period. The figure also shows several anomaly scores which will be scored according to NAB. Any false negative or false positive will contribute negatively to the overall score and only true positives will contribute positively to the overall score.

As shown earlier in Section 6.2.3.1, NAB has the concept of *application profile* which allows for customising the scoring methodology to put more weight on false positives or on false negatives. In this experiment, a profile very similar to the standard profile provided with NAB was used. True positives weight 100%, false positives weight 20% and false negatives weight 100%. Therefore, with the profile used in this research, a single false positive will reduce the overall score by 20%.

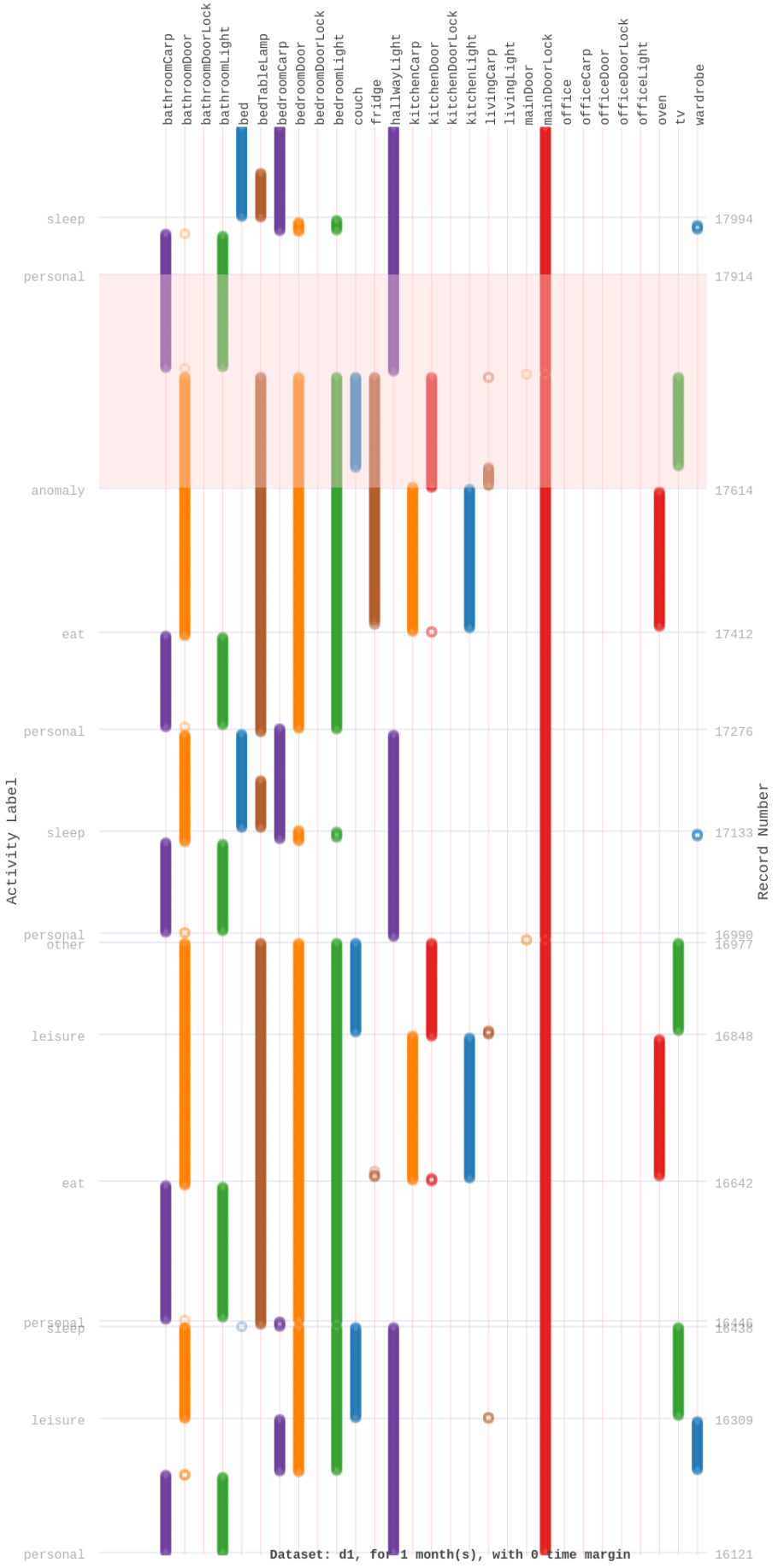


FIGURE 6.5: A portion of dataset d1-1m-0tm that shows the sensors' readings.

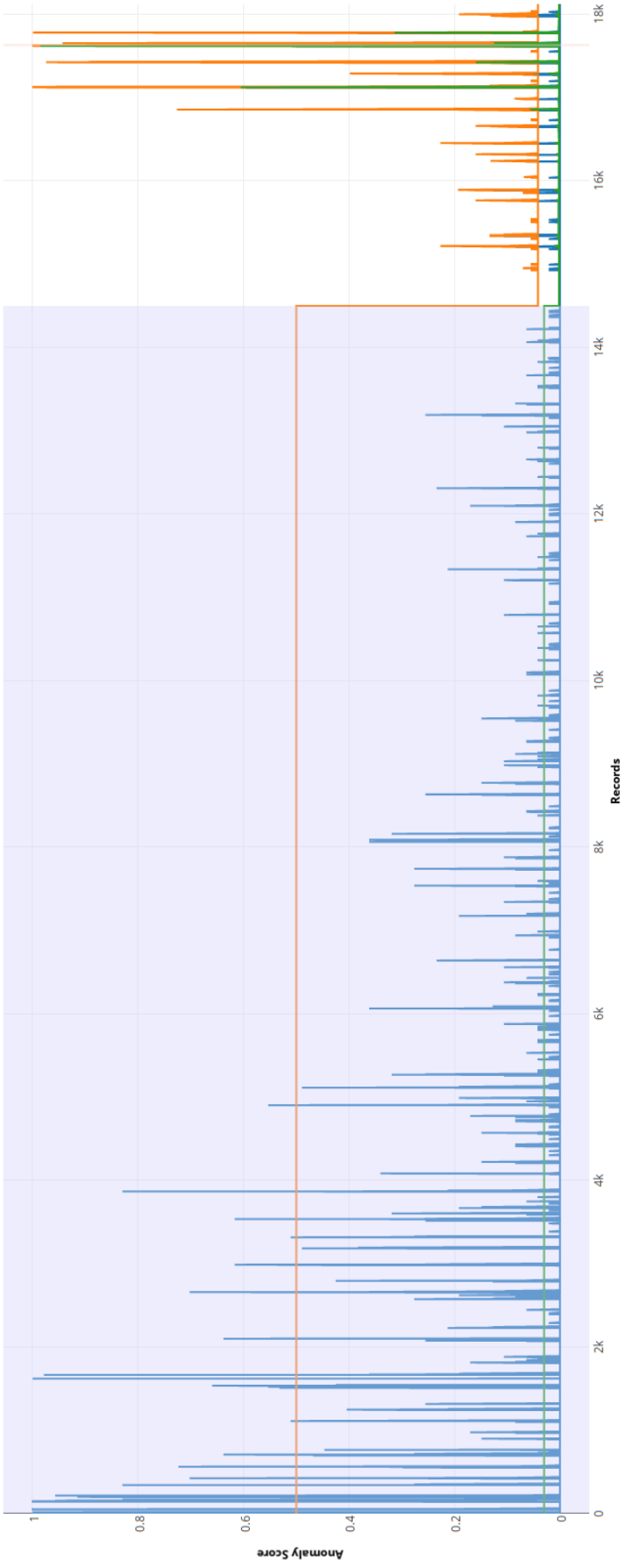


FIGURE 6.6: Scoring an HTM system with HI-SDR encoder on dataset d1-1m-0tm.

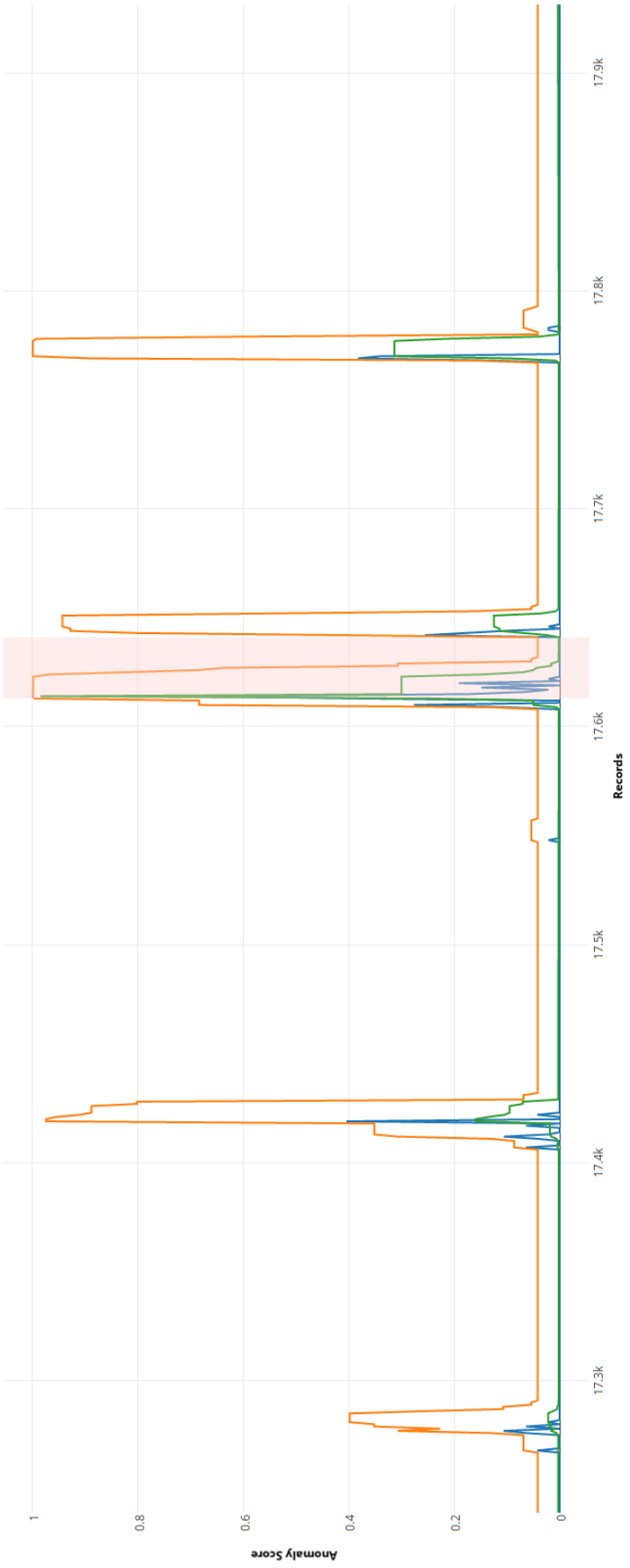


FIGURE 6.7: Part of the last 20% section of the dataset d1-1m-0tm and the anomaly period.

6.3 Results

Several unsupervised anomaly detection techniques and algorithms were tested and this Section presents the results and findings. The techniques can be categorised into four categories: HTM based, Nearest Neighbour based, Cluster and Density based, and Statistics based techniques.

6.3.1 HTM Based

This Section presents the results of running two similar HTM based models with the exception of the encoder used. The first model used the SDR-Category encoder because it produced the best results as shown in Section 5.4.4. The second model uses the HI-SDR encoder and all other aspects of the HTM model are similar.

The tested HTM models have three distinct regions. The encoder region, the SP region and the TM region. The raw anomaly score is the ratio of the wrongly predicted active columns to the total number of columns. In other words, at any given time t, to calculate the anomaly score, count the number of active columns that were predicted at time $t-1$ but are not present in time t and divide it by the total number of all columns. This ratio represents the prediction mistakes of the HTM system.

The raw anomaly score is one of the metrics that can be used to perform anomaly detection and can be useful on its own in certain situations. However, this metric is not the only one used. The anomaly likelihood is another anomaly metric that uses the raw anomaly score to calculate a moving average and predict the probability of seeing this raw anomaly score.

The anomaly metric used in this experiment is the log likelihood of the raw anomaly score. The likelihood anomaly score, explained earlier, can be high and taking the log scale of these values does improve the separation of these anomalies and improve the overall anomaly score.

In Figures 6.6 and 6.7, the three anomaly metrics are show in different colours. The raw anomaly score is coloured blue, the anomaly likelihood is coloured in orange, and the log likelihood is coloured green.

6.3.1.1 SDR-Category Encoder

In this experiment, NuPIC version 0.5.5 and NuPIC Bindings version 0.4.5 were used. Several model parameters for the SP and the TM were used and tested on the forty-two

anomaly datasets generated by the participants. Using one set of parameters, a model is created for each dataset to learn the inhabitant's habits and the performance is scored using NAB (See 6.2.3.1). Finally, the average NAB scores for the forty-two datasets is reported. In the following figures, every data point represents the average NAB score over the forty-two datasets with specific model parameters.

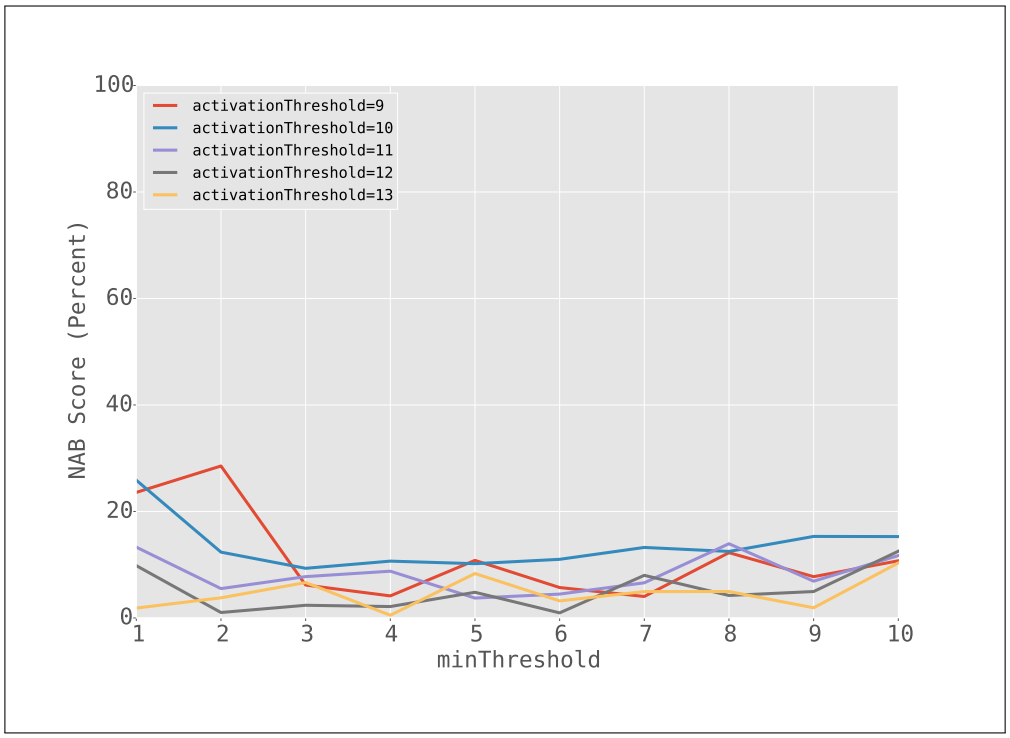


FIGURE 6.8: SDR-Category encoder results with different minThreshold values.

Due to the lack of an optimisation algorithm suitable for HTM systems in high dimensional datasets, several experimental model parameters were used to assess the performance. Figure 6.8 shows the performance of several HTM models with different values for the parameters `activationThreshold` versus the `minThreshold` parameter.

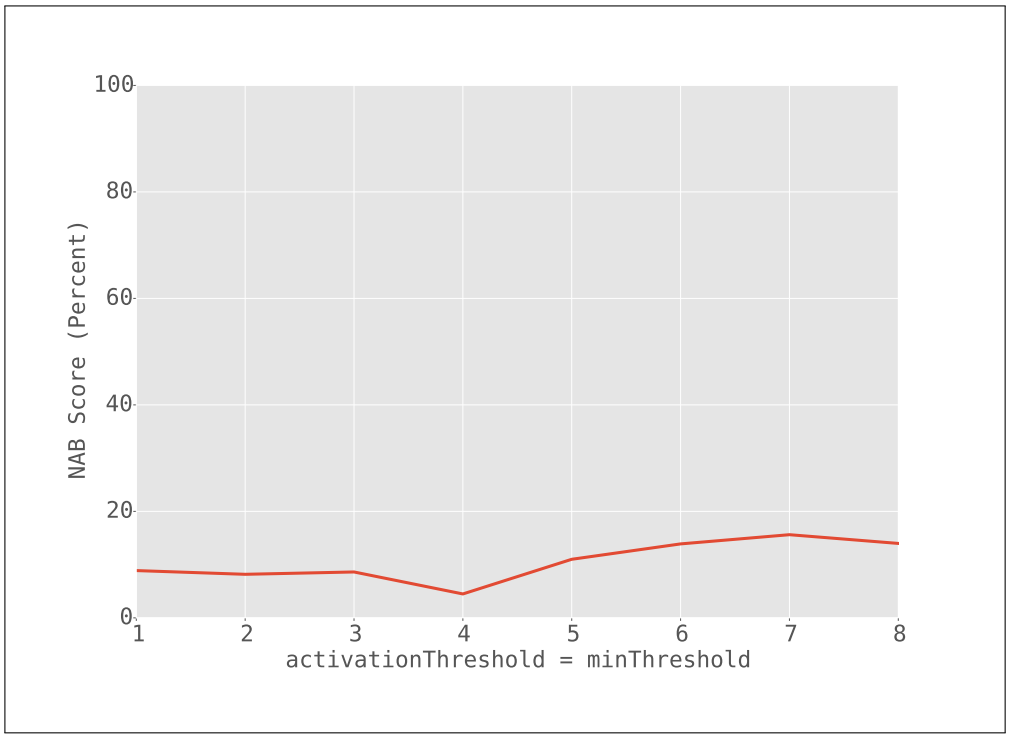


FIGURE 6.9: SDR-Category encoder with activationThreshold values equals to minThreshold.

In the first four passes in Figure 6.8, it seemed that the performance of the models have improved when the values of `activationThreshold` and `minThreshold` are closer together, towards the right-hand side of the graph. Therefore, another pass of performance evaluation was carried out in Figure 6.9 where `activationThreshold` and `minThreshold` are equal to each other. However, the results show no significant improvements.

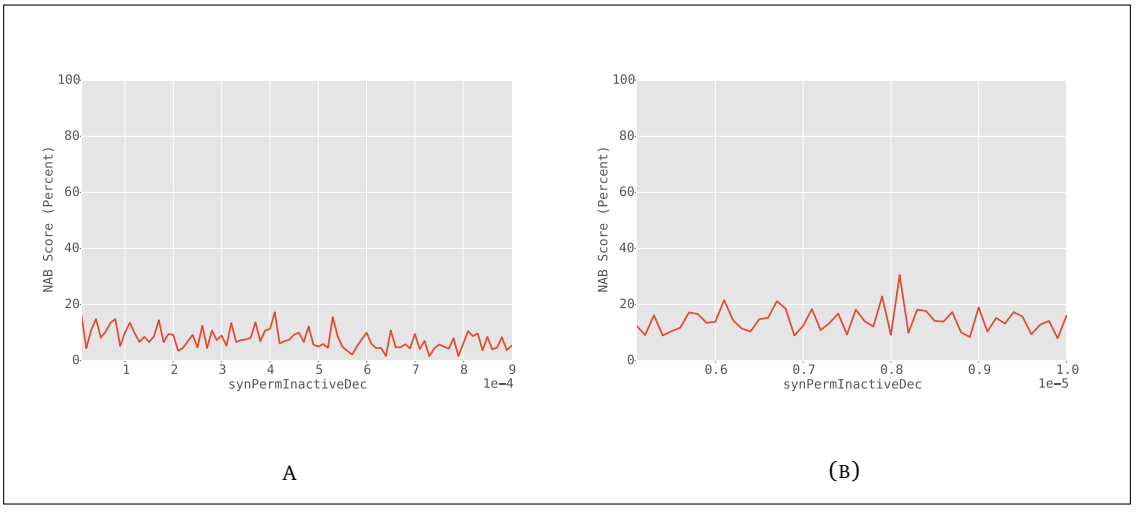


FIGURE 6.10: SDR-Category encoder results and synPermInactiveDec parameter.

Figure 6.10 shows the performance of the HTM model with different values of the parameter `synPermInactiveDec` which specifies the amount of decrement to the inactive synapses. The results of these passes show that around the value of 0.000008 there is an improvement of the model performance. Therefore, in Figure 6.11 two evaluation passes were carried out with the `synPermInactiveDec` parameter set to the best value identified so far (0.0000081). In the first pass in Figure 6.11, the parameter `minThreshold` was set to 10 and in the second pass it was set to 9. In both passes, the parameter `activationThreshold` was evaluated at values ranging from 1 to 15.

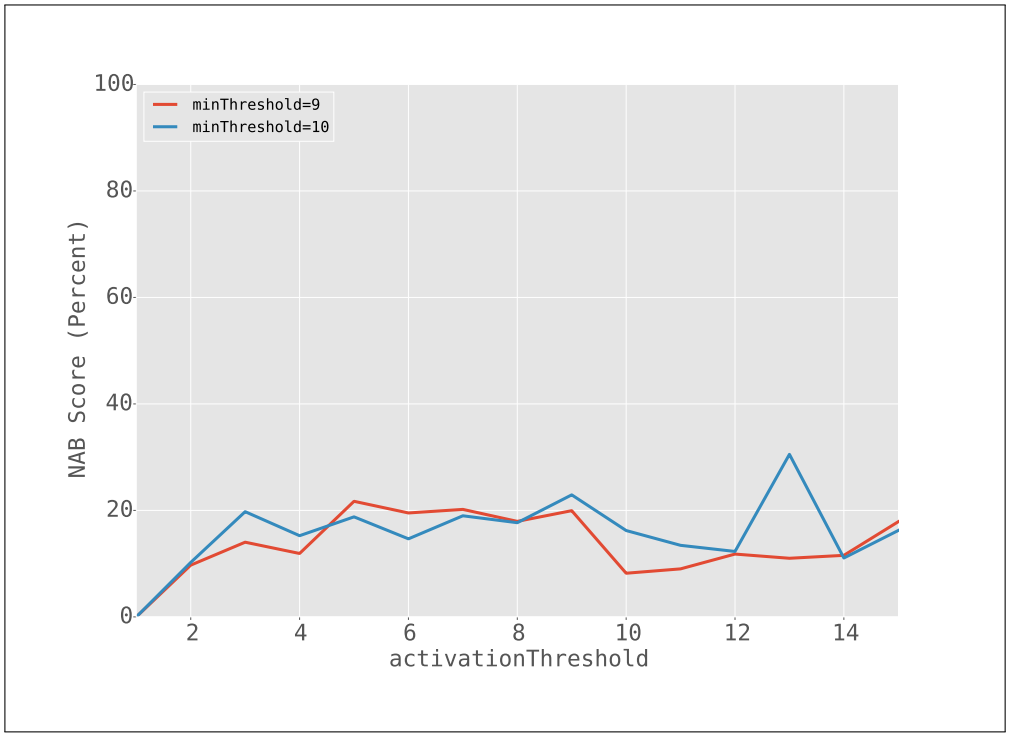


FIGURE 6.11: SDR-Category encoder results with different activationThreshold values.

In Figures 6.12 and 6.13, several performance passes are carried out to test the behaviour of the HTM model with respect to the parameters `activationThreshold` and `minThreshold`. Throughout all of these passes, the models could not exceed the 40% NAB accuracy mark.

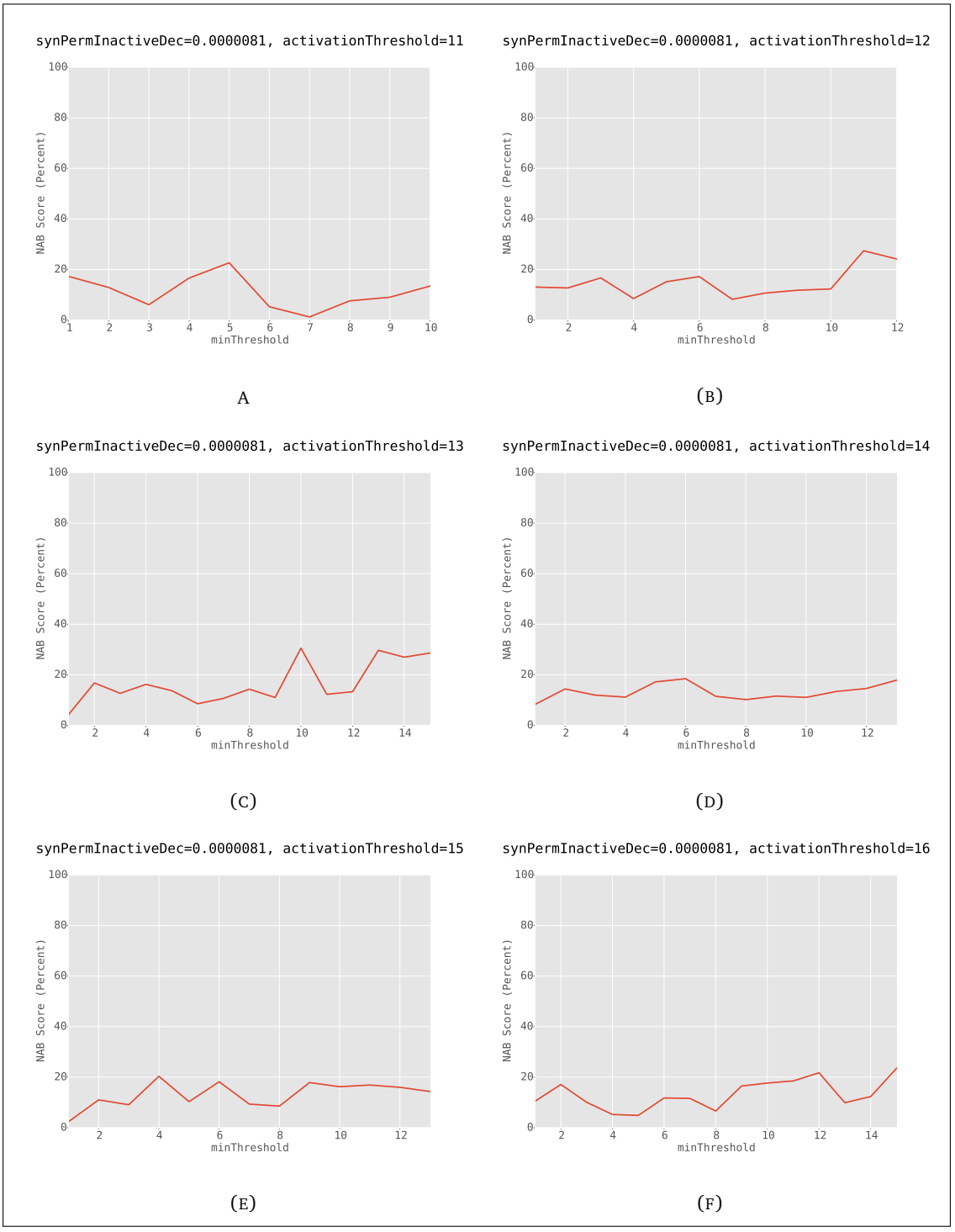


FIGURE 6.12: SDR-Category encoder with activationThreshold values ranging from 11 to 16.

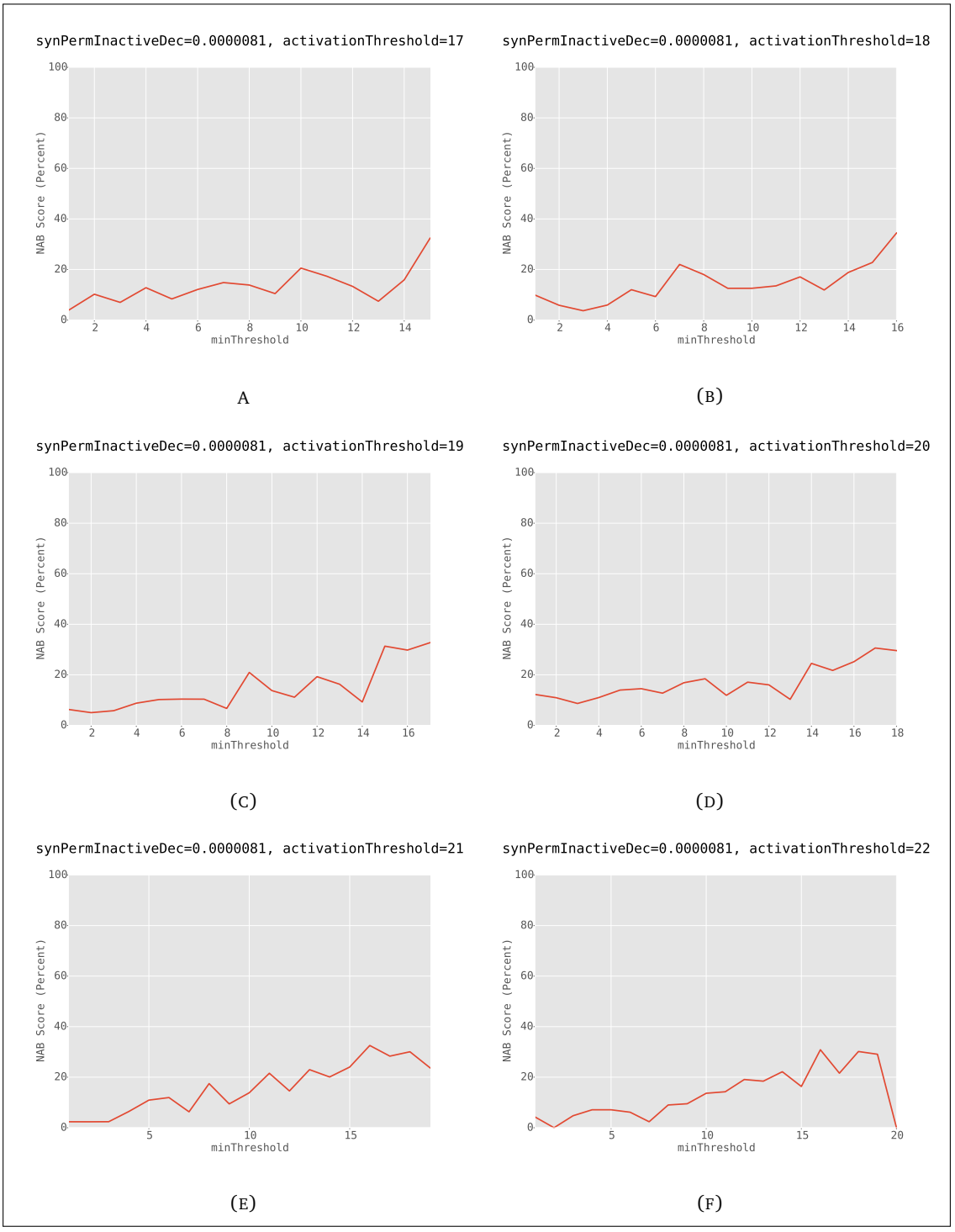


FIGURE 6.13: SDR-Category encoder with activationThreshold values ranging from 17 to 22.

6.3.1.2 HI-SDR encoder

This Section presents the results of using a similar HTM model to the one showed in the previous Section with the exception of the first region in the model. The encoder used in

here is the novel HI-SDR encoder. As explained earlier, the lack of a good optimisation algorithm motivated an empirical experimentation to pick suitable model parameters.

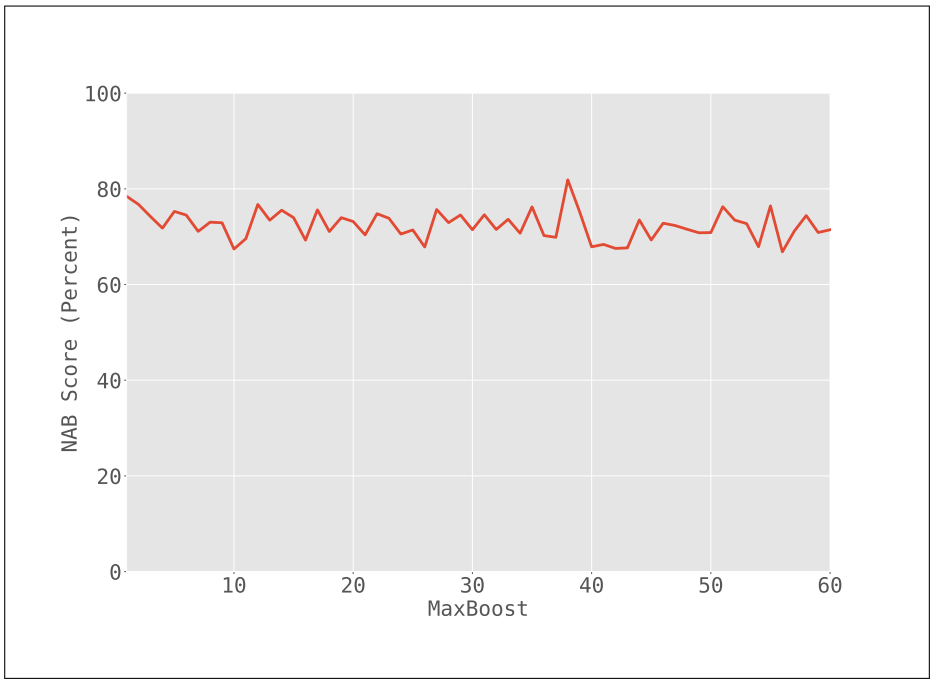


FIGURE 6.14: HI-SDR with different MaxBoost values.

Figure 6.14 shows the behaviour of the model when changing the `MaxBoost` parameter. Similarly, Figure 6.15 shows the performance of the model with different values of the parameter `CellsPerColumns`.

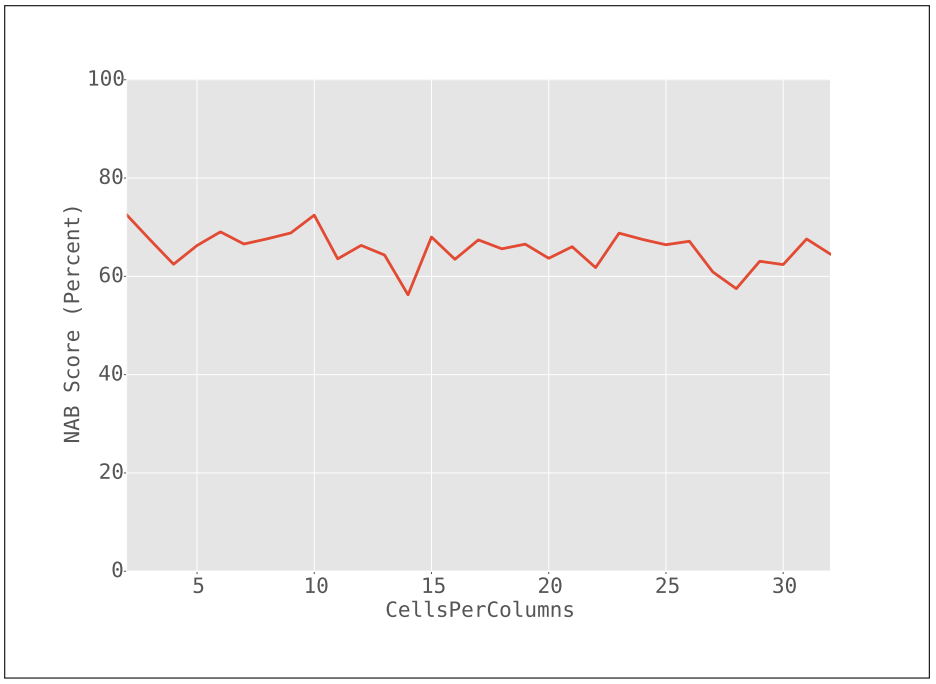


FIGURE 6.15: HI-SDR with different CellsPerColumns values.

In Figure 6.16 the HTM model was evaluated with different `connectedPermanence` values ranging from 0.1 to 0.19.

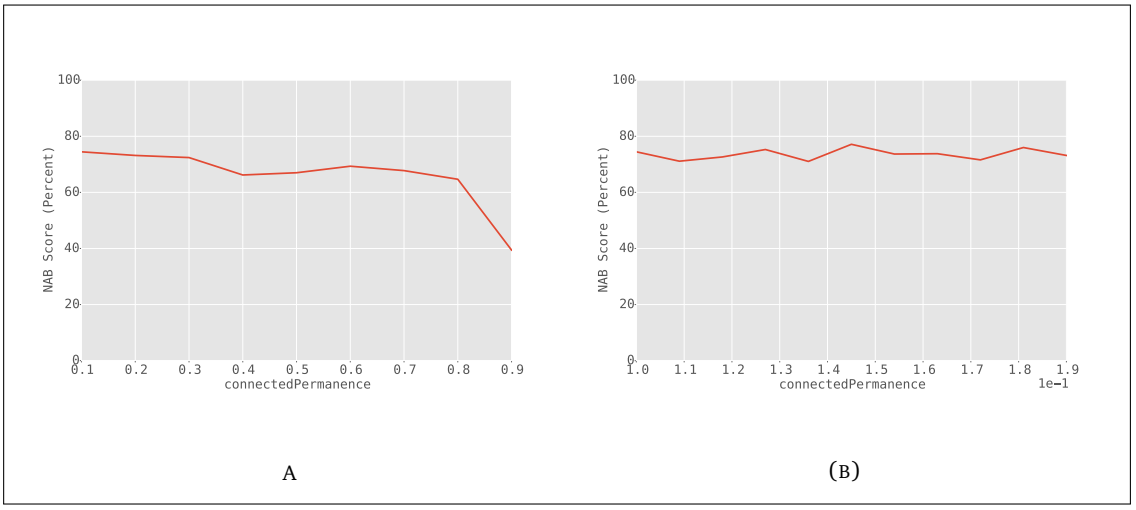


FIGURE 6.16: HI-SDR with different connectedPermanence values.

Figure 6.17 shows the performance of the model when changing the parameter `numActiveColumnsPerInhArea`. The NAB score shows a steady increase when increasing the number of active columns per inhibition area, up to the 80% mark. Since this parameter is directly related to the number of active bits at any given moment, it is expected to have low performance with low number of active bits because there will not be enough space for the model to store the state of the world in. However, after a certain value, the performance steadily decreases. In this model the total number of bits n is 600. The interval where the model produced the best results was between 20 to 40 bits. The ratio of active bits in this interval is around 3% to 6% which aligns well with the HTM literature recommendation of the number of active bits for a given model.

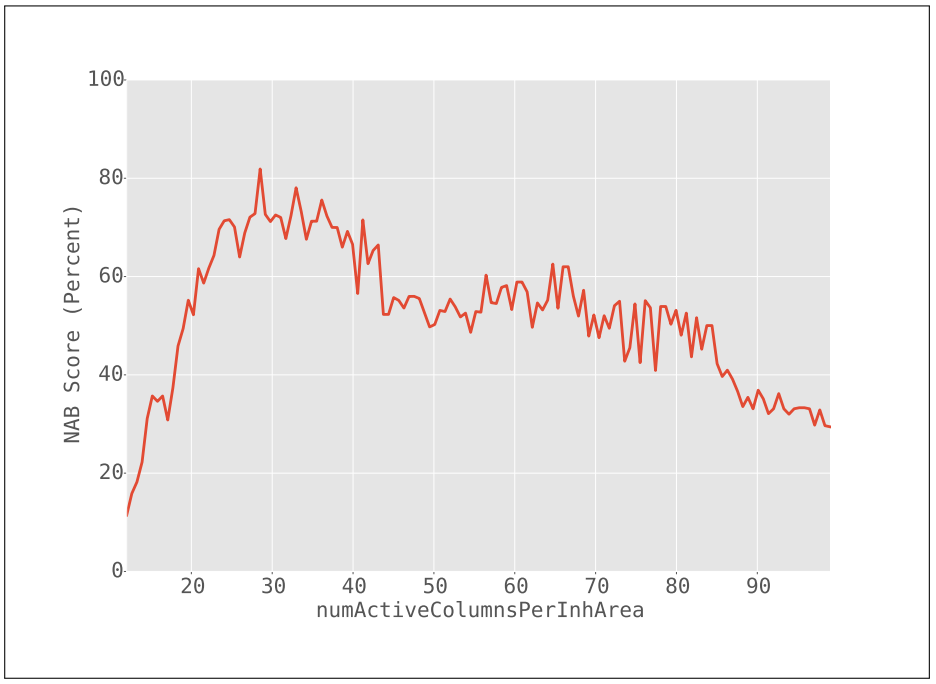


FIGURE 6.17: HI-SDR with different numActiveColumnsPerInhArea values.

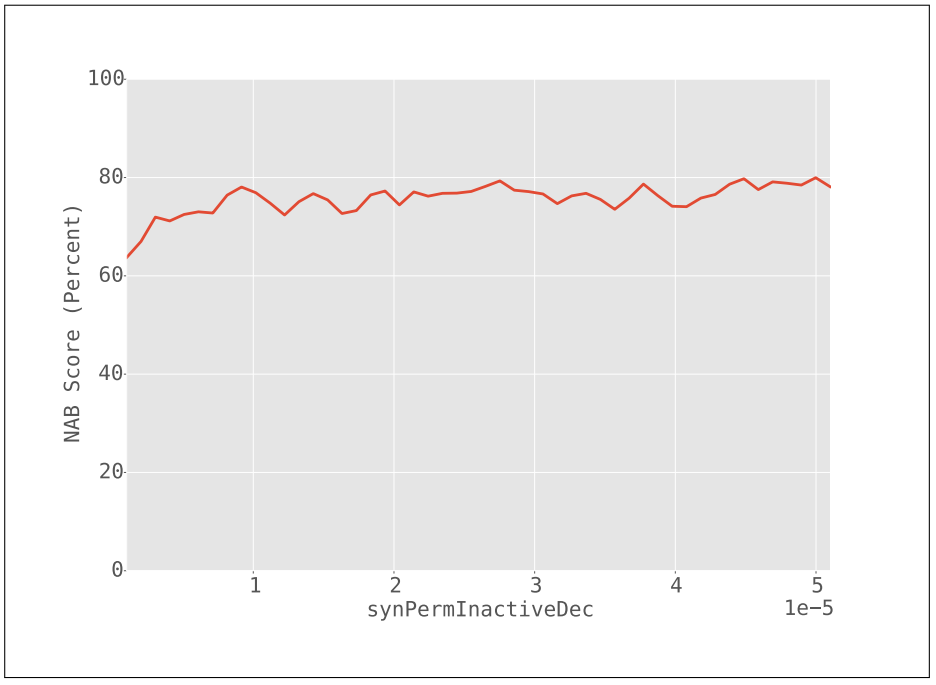


FIGURE 6.18: HI-SDR with different synPermInactiveDec values.

6.3.1.3 SDR-Category Encoder versus HI-SDR

In this Section, the results of using identical sets of model parameters for two HTM models with different encoders. The idea in this Section is to eliminate the influence

of the model parameters on the model results and narrow down the difference between the two models to the type of encoder used.

```
SpatialPooler(
                inputDimensions=(3590,),
                columnDimensions=(2048,),
                synPermConnected=0.1,
                synPermActiveInc=0.05,
                synPermInactiveDec=0.05015,
                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.

Using the model parameters shown in Listing 6.1, the average NAB score of the HTM model with SDR-Category encoder on the forty-two datasets was 12.46%. Using the same model parameters with an HTM model equipped with the novel HI-SDR encoder, the average NAB score was 36.88%.

From our empirical experimentation of finding the best model parameters for HI-SDR encoder, the parameters shown in Listing 6.2 produced the best results. The average NAB score was 81.89%. Using the same model parameters with an HTM model equipped with the SDR-Category encoder, the average NAB score was 2.14%.

```
SpatialPooler(
                inputDimensions=(600,),
                columnDimensions=(600,),
                synPermConnected=0.1,
                synPermActiveInc=0.05,
                synPermInactiveDec=0.00001,
                globalInhibition=True,
                numActiveColumnsPerInhArea= 38,
                maxBoost=38.0,
                potentialPct=0.8),

TemporalMemory(
                columnDimensions = (600,),
                cellsPerColumn=2,
```

```
initialPermanence=0.21,
connectedPermanence=0.20,
minThreshold=4,
maxNewSynapseCount=20,
permanenceIncrement=0.5,
permanenceDecrement=0.001,
maxSegmentsPerCell=128,
maxSynapsesPerSegment=32,
activationThreshold=9)
```

LISTING 6.2: The best parameters found for HI-SDR models.

6.3.2 Nearest Neighbour Based

In this Section, several nearest neighbour models are evaluated using RapidMiner[2] version 7.4 and the anomaly detection[3] extension version 2.3. Each model's anomaly score is normalised to the range 0.0 to 1.0. The higher the value is, the higher the likelihood of an anomaly occurring.

Starting with k-Nearest Neighbour (k-NN), the algorithm requires the parameter k to be set. Testing the algorithm with different values of k and their results are shown in Section 6.3.2 for values ranging from 2 to 7. Since several features are in the datasets, using Principal Component Analysis (PCA) as a preprocessing step was also tested and the results are shown in Section 6.3.2 for values for k ranging from 2 to 13. Reducing the dimensionality of the input using PCA seems to have improved the results of the algorithm.

[2]https://rapidminer.com/
[3]http://madm.dfki.de/rapidminer/anomalydetection

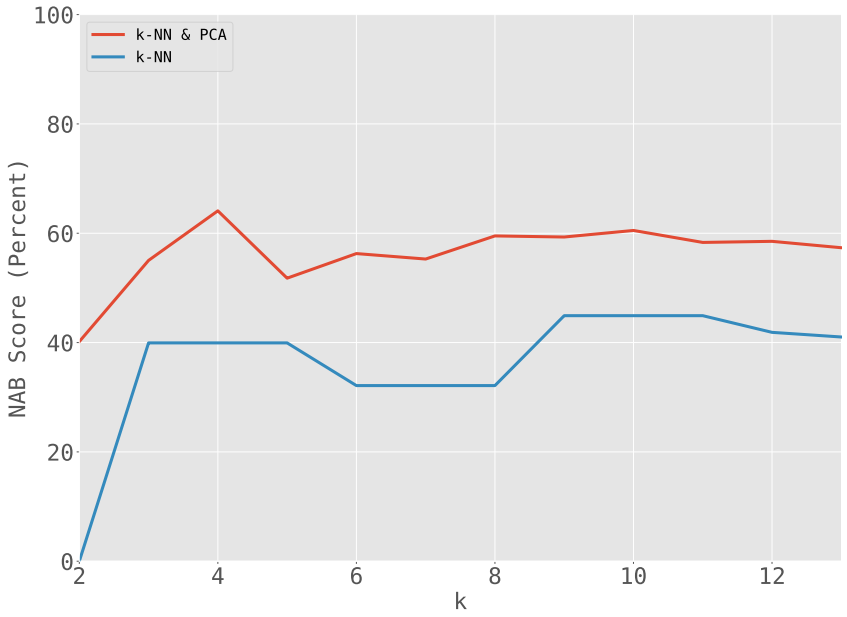


FIGURE 6.19: k-NN model results.

Table 6.4 shows the results of the Local Outlier Factor (LOF) algorithm with PCA as a preprocessing step and without it. The implementation of the LOF algorithm in the anomaly detection extension for RapidMiner runs several values of k and uses the best score. Unlike the previous algorithm, using PCA does not seem to improve the results of the model.

TABLE 6.4: LOF model results.

LOF & PCA		**LOF**	
k	NAB Score	k	NAB Score
2-10	11.56%	2-10	15.0%

Section 6.3.2 shows the results of the Connectivity Based Outlier Factor (COF) with PCA and without it. Several k values were tested for both models. The application of PCA seems to have improved the results slightly.

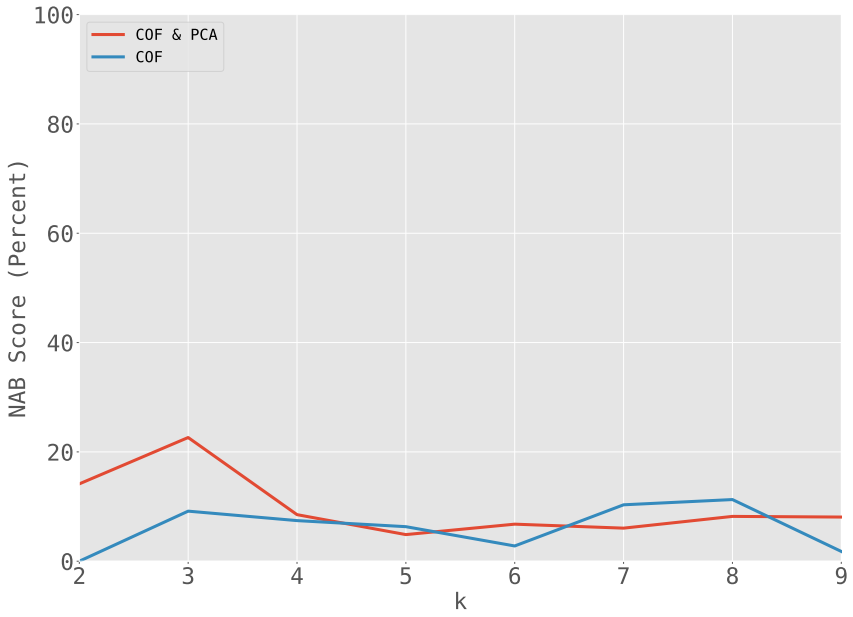


FIGURE 6.20: COF model results.

Two models using the Local Correlation Integral (LOCI) algorithm results are shown in Table 6.5 one with PCA as a preprocessing step and the other model without a dimensionality reduction algorithm. The algorithm seems to achieve better performance without the dimensionality reduction step.

TABLE 6.5: LOCI model results.

LOCI & PCA	**LOCI**
NAB Score	NAB Score
32.89%	41.13%

The Local Outlier Probability (LoOP) results are shown in Section 6.3.2 with PCA and without it. With low values of k the model was not able to detect the anomalies without the dimensionality reduction step. However, with values of $k > 6$ the application of PCA seems to increase the performance of the model.

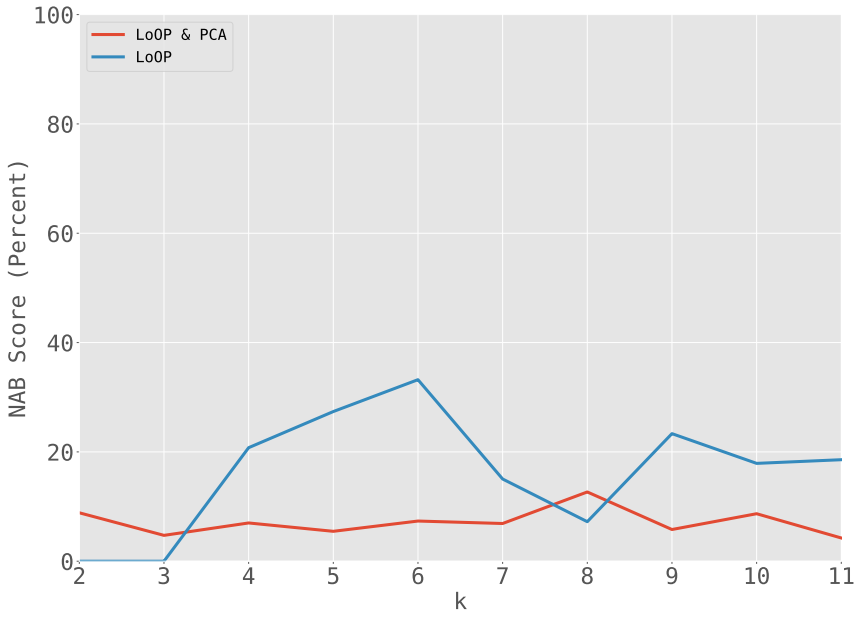


FIGURE 6.21: LoOP model results.

The Influenced Outlierness (INFLO) algorithm, in similar fashion, was run with PCA and without it and its results are shown in Section 6.3.2. The results are slightly better without the dimensionality reduction using PCA.

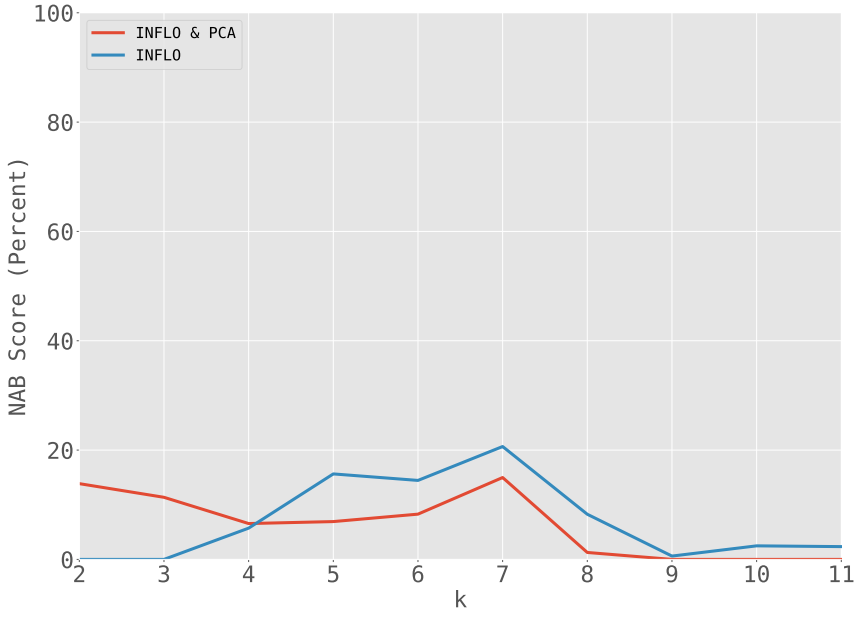


FIGURE 6.22: INFLO model results.

6.3.3 Cluster and Density Based

In this Section, several clustering and density based models are evaluated using RapidMiner version 7.4 and the anomaly detection extension version 2.3. Similar to the previous Section, every model's anomaly score is normalised to the range 0.0 to 1.0. The cluster based algorithms require a clustering algorithm to be used in conjunction with an anomaly detection algorithm.

Using the Cluster Based Local Outlier Factor (CBLOF) algorithm, the results of two models are shown in Table 6.7. The difference between the two models is the choice of the clustering algorithm used. The k-means and its extension, the X-means algorithms were chosen in this Section. The X-means algorithm tries to overcome the issue of choosing an appropriate k value for the clustering algorithm. Thus, only one model result is presented in conjunction with the X-means algorithm. The results shows very little difference between the best k-means score and the X-means score with the CBLOF algorithm.

TABLE 6.6: CBLOF model results.

k-means		**X-means**	
k	NAB Score	k	NAB Score
2	7.40%	2-60	14.79%
3	10.50%		
4	15.50%		
5	16.07%		
6	9.80%		
7	15.60%		
8	15.42%		
9	13.18%		

The Local Density Cluster Based Outlier Factor (LDCOF) results in Table 6.7 tell a similar story to the previous algorithm. The X-means score is very close to the best score of the k-means model.

TABLE 6.7: LDCOF model results.

k-means		**X-means**	
k	NAB Score	k	NAB Score
2	9.78%	2-60	14.79%
3	14.90%		
4	12.05%		
5	9.07%		
6	5.78%		

The Clustering-based Multivariate Gaussian Outlier Score (CMGOS) can be categorised under the statistical based anomaly detection algorithm because it uses Gaussian model to determine the anomalies. Table 6.8 shows the results of using CMGOS with k-means and X-means.

TABLE 6.8: CMGOS model results.

k-means		**X-means**	
k	**NAB Score**	**k**	**NAB Score**
2	10.58%	2-10	18.66%
3	13.93%		
4	3.31%		
5	0.0%		

DBSCAN algorithm was not available for the anomaly detection extension in RapidMiner. The Scikit-Learn DBSCAN implementation version 0.18.1 [4] was used in this experiment. Using PCA as a preprocessing step did improve the results slightly as shown in Table 6.9.

TABLE 6.9: DBSCAN model results.

DBSCAN & PCA	**DBSCAN**
NAB Score	NAB Score
27.14%	23.72%

6.3.4 Statistics Based

Similar to the previous Sections, RapidMiner version 7.4 and the anomaly detection extension version 2.3 were used to assess two statistical anomaly detection algorithms.

[4] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

TABLE 6.10: HBOS model results.

HBOS & PCA	**HBOS**
NAB Score	NAB Score
61.90%	28.51%

The Histogram Based Outlier Score (HBOS) algorithm results are shown Table 6.10 with the application of PCA and without applying it. The model results did improve significantly with the PCA application as a dimensionality reduction step of the features. The results of the Robust Principal Component Analysis Anomaly Score (rPCA) are shown in Table 6.11.

TABLE 6.11: rPCA model results.

NAB Score
9.50%

6.4 Discussion

After assessing the anomaly detection performance of several algorithms and techniques, the best scores identified by the experiment design, explained in Section 6.2.3, for each algorithm and techniques are summarised in Figure 6.23. The Figure shows that the HTM based model with the novel HI-SDR encoder is capable of competing and exceeding the performance of the state-of-the-art algorithms. The k-NN and the HBOS models, with dimensionality reduction preprocessing step, produced the second best performance with a slight edge for the k-NN model.

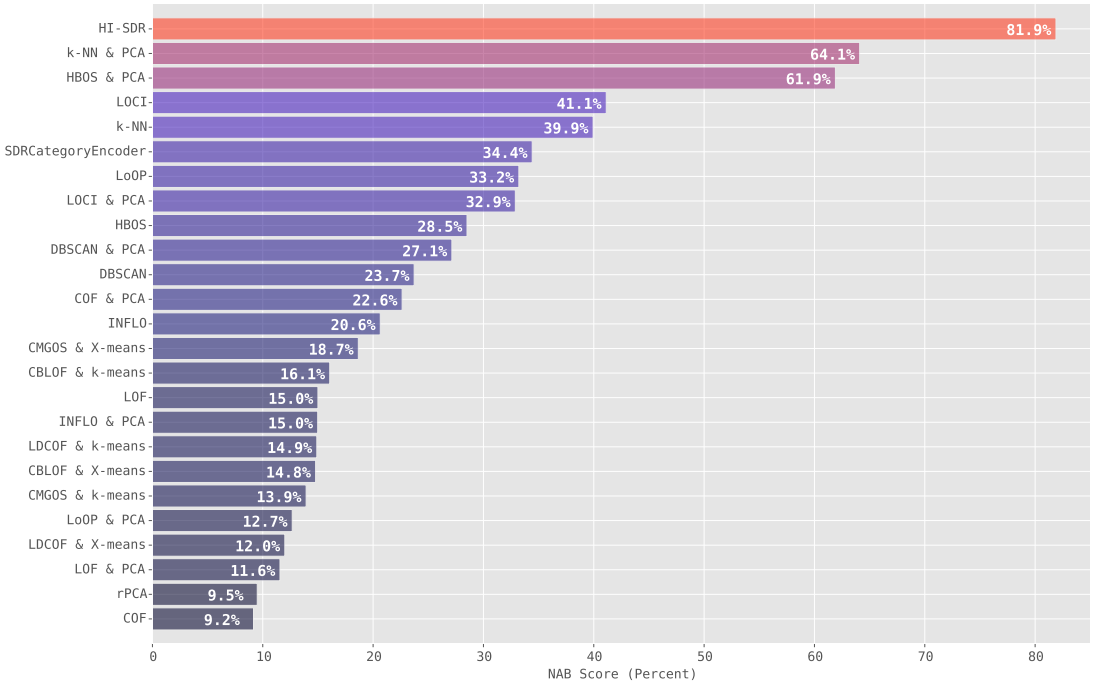


FIGURE 6.23: The results of all evaluated models.

Goldstein & Uchida (2016) carried out a study to evaluate several unsupervised anomaly detection algorithms and models on ten datasets from various domains such as in image dataset of breast cancer, handwriting dataset, a speech accent dataset, and others[5]. Their findings suggest that algorithms that are designed to detect local anomalies such as LOF and COF produce poor results especially when the dataset only contains global anomalies. On the other hand, global anomaly detection methods such as k-NN and HBOS produce better results and on datasets with local anomalies their performance is average. Thus, they recommend the use of global anomaly detection algorithms unless the dataset strictly contains local anomalies. The results shown in Figure 6.23 show similar findings. Indeed, the k-NN algorithm and the HBOS algorithm produced the best results other than the HTM model with the HI-SDR encoder. One interesting finding is that the LOCI algorithm was not recommended by Goldstein & Uchida (2016) for global anomalies. However, in this study the algorithm is the fourth best performing algorithm. This is an indication that the forty-two datasets generated by OpenSHS are diverse.

Another comprehensive study of unsupervised anomaly detection conducted by Campos *et al.* (2016) on large collection of datasets with different data types from various domains concluded that the k-NN algorithm is one of the state-of-the-art algorithms for

[5]All datasets are available at `http://dx.doi.org/10.7910/DVN/OPQMVF`

unsupervised anomaly detection. The findings of this study do agree with their conclusion as k-NN is the second best performing algorithm on the smart home datasets.

As for the HTM based algorithms, the novel HI-SDR encoder did significantly improve the performance over a standard HTM model with an SDR-Category encoder. The standard HTM model did produce acceptable results compared to other unsupervised anomaly detection techniques. However, the HI-SDR encoder provided the HTM model with significant increase from 34.42% NAB score using the SDR-Category encoder to 81.89% (an increase of 137.91%).

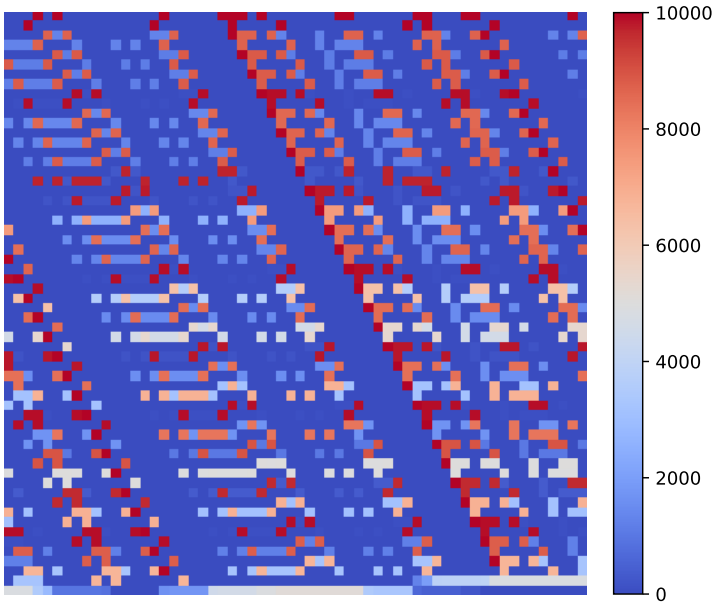


FIGURE 6.24: The first 10,000 records of the d4-2m-10tm dataset encoded with SDR-Category encoder and with model parameters A.

To analyse the reason behind this significant improvement in performance, a closer look at what is happening in the HTM model over time is needed. For example, looking at the first 10,000 records of the dataset `d4-2m-10tm` and how the SDR-Category encoder encodes these records over time should provide an insight and highlight the main issue with the encoder. The heat map in Figure 6.24 clearly shows that a large number of active bits in the generated SDRs are activated at each record. This high number of similar signatures of the inputs makes it hard for the HTM model to distinguish the records. It is worth noting that the SDRs are actually one dimensional array and are shown here as two-dimensional matrices just for visualisation purposes.

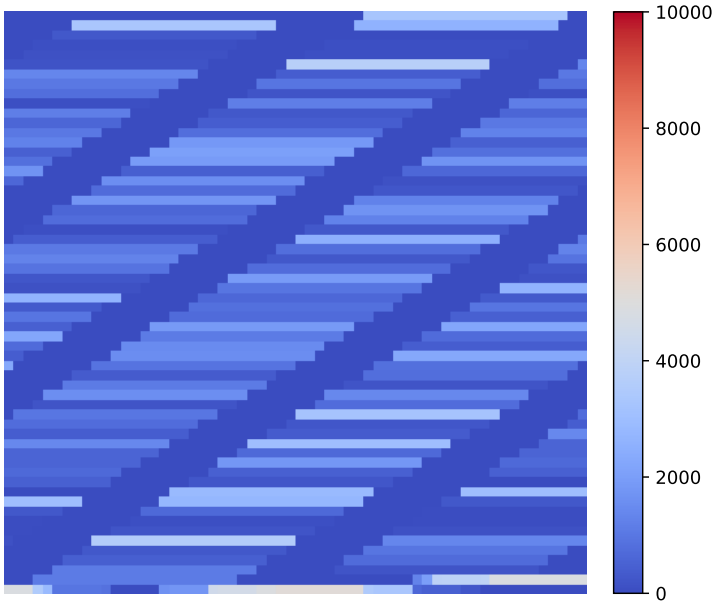


FIGURE 6.25: The first 10,000 records of the d4-2m-10tm dataset encoded with HI-SDR encoder and with model parameters A.

Contrast the previous heat map with the one generated by the HI-SDR encoder shown in Figure 6.25. The active bits' activities are distributed across the available space in a more even manner. The highest active portion of the SDR are in the 5,000 range while in the previous example, they were in the 10,000 range. This issue was discussed in detail in Section 5.4.4.

One of the important points raised by Campos *et al.* (2016) when evaluating unsupervised anomaly detection is the issue of choosing appropriate model parameters. A misconfigured model could perform poorly even if the algorithm or the technique used is appropriate to the task. To rule out this issue when comparing the two HTM models in this study, in Section 6.3.1.3 two identical sets of parameters where used in two models. One with the SDR-Category encoder and the other with the novel HI-SDR encoder. Let us refer to the model parameters shown in Listing 6.1 as A and model parameters shown in Listing 6.2 as B. The set of model parameters A were suggested by a particle swarm optimisation (PSO) technique commonly used with HTM models to arrive at good model parameters. The swarm process is mainly used with classification problem in HTM literature and this may explain the poor results of the suggested model parameters A. In the HTM literature there is a recommended set of model parameters for anomaly detection but it is best suited for datasets with a single scalar feature. Therefore, that set of parameters did not perform well in this dataset which has high dimensional categorical features. The model parameters B are the best parameters identified in this study for the HI-SDR encoder.

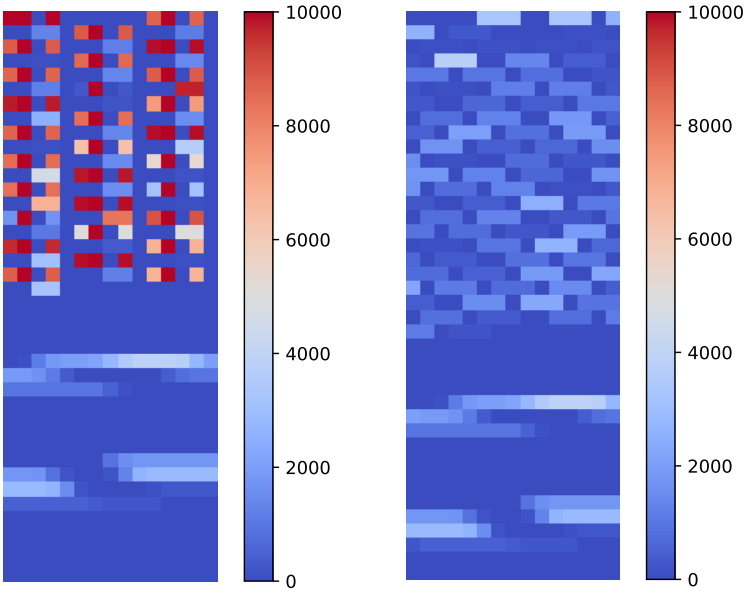
Table 6.12 shows a comparison of the NAB scores of the two models using the two model parameters sets A and B. In both sets of parameters, the model equipped with HI-SDR encoder excels.

TABLE 6.12: SDR-Category encoder and HI-SDR encoder results using two identical sets of parameters.

Encoder	**Parameters** A	**Parameters** B
SDR-Category encoder	12.46%	2.14%
HI-SDR	36.88%	81.89%

To gain more insight on why the results are better when using the HI-SDR encoder, let us look at the resulting SDRs of the encoders, what is happening in the SP region, and what is happening in the TM region.

The previously shown Figure 6.26 illustrates the generated SDRs by the SDR-Category encoder and the HI-SDR encoder using the model parameters A.



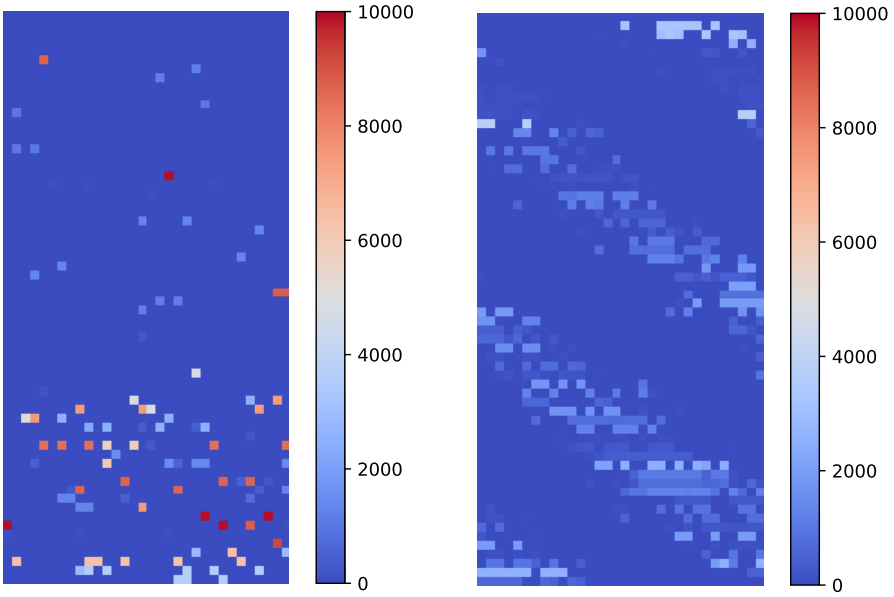
(A) SDR-Category encoder (B) HI-SDR encoder

FIGURE 6.26: The first 10,000 records of the d4-2m-10tm dataset encoded with both encoders with model parameters B.

Figure 6.27 shows the two models with parameters B. Very similar outcome is observed and illustrated in the generated SDRs. The HI-SDR encoder manages to produce more evenly distributed active bits across the available space of the SDR which gives the subsequent regions of the HTM model the ability to easily distinguish the input. While

the SDR-Category encoder reuses and activates similar active bits which leads to similar inputs to the subsequent HTM regions.

The SP region is the next region in the HTM system and it is directly affected by the SDRs generated by the encoders. Figure 6.27a shows the SDR-Category encoder with model parameters A. The heat map shows very similar features, high activities are concentrated in very small parts of the SP.

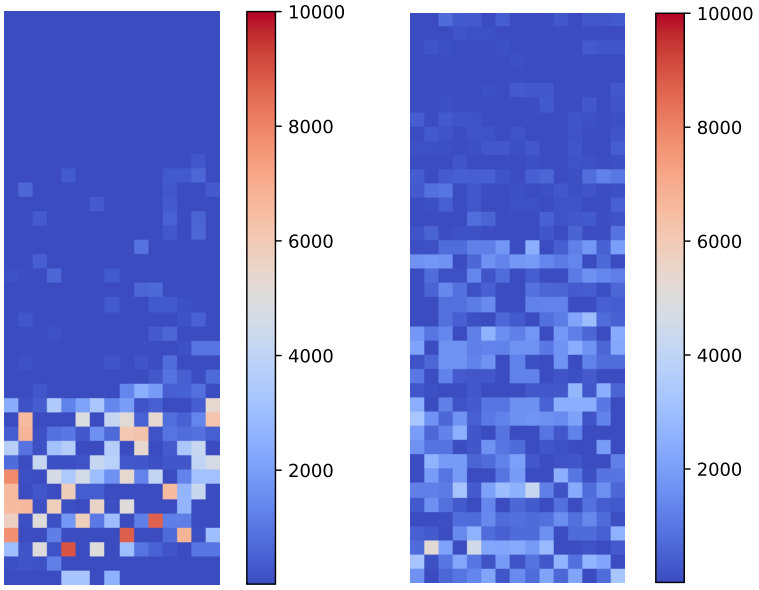


(A) SDR-Category encoder (B) HI-SDR encoder

FIGURE 6.27: The SP with both encoders and with model parameters A.

Contrasting that with what is shown in Figure 6.27b which shows an evenly distributed activities the available SP region. This indicates that different parts of the SP are learning much more characteristics about the input than the previous model.

Figure 6.28 shows the two models with the model parameters B and again, very similar findings. High and concentrated activities of the SP region with the SDR-Category encoder and evenly spread activities with the HI-SDR encoder.



(A) SDR-Category encoder (B) HI-SDR encoder

FIGURE 6.28: The SP with both encoders using model parameters B.

To analyse what is happening in the TM, one useful metric is the number of bursting columns and their behaviour over time. Figure 6.29 shows the bursting columns for the first 10,000 records of the dataset d4-2m-10tm of the model equipped with the SDR-Category encoder. The number of the bursting columns is 6061 columns. While the HI-SDR equipped model shown in Figure 6.30, seems very active with 78094 bursting columns.

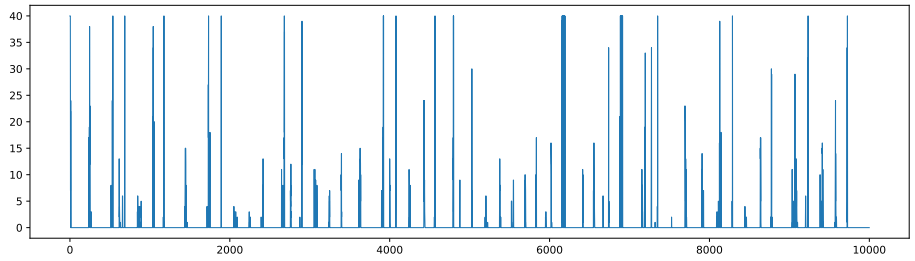


FIGURE 6.29: The TM with SDR-Category encoder and with model parameters A.

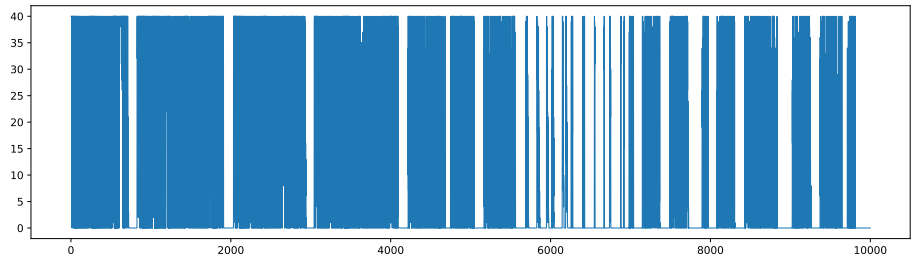


FIGURE 6.30: The Temporal Memory with HI-SDR encoder and with model parameters A.

Figure 6.31 shows the bursting columns with SDR-Category encoder and the number of the bursting columns is 2305. This model seems very relaxed and not alerted to the incoming data. The reason for this is that the SDRs generated by SDR-Category encoder look similar. Thus, the model cannot distinguish between the different input records and becomes numb. On the other hand, Figure 6.32 shows the HI-SDR encoder bursting columns which are totalling 4327 columns. The model shows a smooth decrease in the number of bursting columns over time and yet never going to be completely relaxed. There seems to be a trade-off between how relaxed and sensitive the model is and its capabilities to detect anomalies.

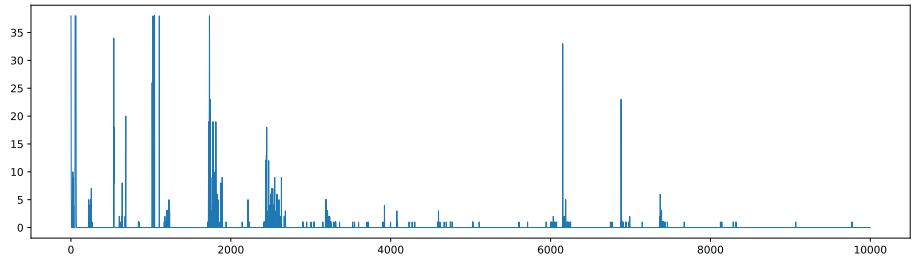


FIGURE 6.31: The TM with SDR-Category encoder and with model parameters B.

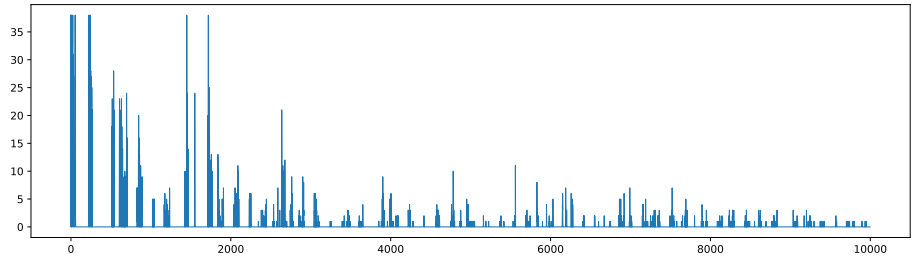


FIGURE 6.32: The Temporal Memory with HI-SDR encoder and with model parameters B.

6.5 Summary

In this Chapter, the design and methodology of the whole experiment were explained. The virtual smart home (that was built using OpenSHS) was presented and a listing of all the sensors and smart devices along with the type of data they generate. The smart home was equipped with twenty-nine binary sensors including various smart devices. Also, the rationale behind using binary sensors was explained.

The forty-two datasets that were generated using the virtual test bed were presented. Each of the seven participants were trained and guided by the researcher to use OpenSHS and then they were asked to perform several tasks to ensure a good level of familiarity with the simulation tool. After making sure that the participants are comfortable with OpenSHS, the actual simulations took place. The participants were asked to simulate four contexts: 'morning-weekday', 'morning-weekend', 'evening-weekday', and 'evening-weekend' contexts. After that, each participant was asked to perform an anomaly according to their own definition and patterns. The type of anomalies that the participant decided to perform and simulate are explained along with the OpenSHS parameters used to aggregate the final datasets. The forty-two datasets totalled 2,743,897 records worth of data.

A detailed explanation of the experiment design and the evaluation pipeline for all the tested algorithms was presented and discussed. Each experiment starts by reading one dataset out of the forty-two datasets and feeding the data to the anomaly detection model. 80% of the dataset is used to let the model learn from without scoring. The last 20% of the dataset is used to score the performance of the model. Most of the tested anomaly detection models score was a scalar value which imposed a challenge for evaluation. The challenge was figuring out what is the best threshold to choose to recognise a reading as an anomaly or not. To overcome this issue and to make the evaluation process fair for all models, an optimisation algorithm (Twiddle) was used and several iterations were performed for each dataset to find the best threshold possible for that dataset. The best score was the final score reported for that dataset. Finally, an average for all the datasets scores was reported as the model overall accuracy in detecting the anomalies.

The results of several unsupervised anomaly detection algorithms and techniques were reported on the participant's generated datasets. The models were grouped by the primary technique used in the model. The experiments covered HTM based, nearest neighbour based, cluster and density based and statistics based techniques. A detailed discussion of these results shows that previously conducted studies about unsupervised

anomaly detection techniques on different datasets produced results similar to the results found by this experiment.

As for the HTM based anomaly detection models, without the novel HI-SDR encoder, they produced moderate results that could not compete with the state-of-the-art anomaly detection algorithms. However, with the HI-SDR encoder, the HTM models were able to compete and exceed these algorithms. The HI-SDR encoder scored 81.9% accuracy, on the forty-two datasets, with 17.8% increase in accuracy compared to the k-NN algorithm and 47.5% increase over the standard CLA encoders.

The Principal Component Analysis (PCA) algorithm was used as a preprocessing step and this step improved the results of some of the algorithms. For example, the k-NN algorithm scored 39.9% accuracy without PCA and scored 64.1% accuracy with PCA. Similar results were achieved by the Histogram Based Outlier Score (HBOS) algorithm which scored 28.5% accuracy without PCA and 61.9% with PCA as a preprocessing step.

One possible improvement for the HTM based models to develop an optimisation algorithm to tune its parameters for anomaly detection problems. A Particle Swarm Optimisation (PSO) technique is usually used in the literature, but it is more suitable for classification and prediction problems.

Chapter 7

Conclusions and Future Work

Smart homes are becoming more sophisticated and complicated due to the number of smart devices that are added to them and the different tasks that these devices are performing. Nowadays, with the realisation of the IoT vision, it is becoming more important for the smart home inhabitant to take control and manage these devices. The IoT vision in the smart home faces several challenging issues such as security, privacy, elderly care and context-awareness. Anomaly detection can play an important role in solving some of these issues.

The literature review of detecting anomalies in the smart home domain showed a lack of good datasets that are focused on anomaly detection problems. The existing research projects in this domain, usually focus on the classification and recognition of ADLs for the inhabitants and there are many real-world datasets generated from actual smart homes in the literature. However, few research projects focused on anomaly detection problems.

The habits and daily patterns of the smart home inhabitants are different and ever-changing. This fact imposed an interesting challenge to this research. Proposing an anomaly detection technique based on supervised machine learning algorithms is rather limiting and not generalisable. Thus, unsupervised machine learning algorithms and techniques were the target of this research due to their ability to learn from the data without the need for training labels in the dataset.

The HTM theory introduces an overarching theory that proposes the existence of a common algorithm performed across all regions of the brain's neocortex. The algorithmic implementation of the theory, the CLA, proposes several regions that realise the HTM theory and make it applicable to real-world problems. The first region is the encoder

region which is responsible of transforming the streaming input data to a sparse representations called SDRs. The next region is the SP which learns the spatial features of the SDRs and pass this new learned representation to the next region, the TM. The TM is responsible for learning the transitions from one SDR to the next, thus learning the temporal features of the streaming data. All of these regions are biologically inspired by the recent findings in Neuroscience.

The HTM models can learn the state of the world and can be used to perform classification, prediction and anomaly detection problems. Although extracting what the HTM model has learned is currently done with statistical approaches and not by biologically inspired techniques. Yet, the results of detecting anomalies seems promising.

This work tackled the lack of good representative datasets aimed at anomaly detection and developed OpenSHS, a new hybrid, open-source, cross-platform, 3D smart home simulator for dataset generation. OpenSHS reduces the time and effort for the researcher and the participants and streamlines the process of generating simulated smart home datasets. Using this tool, forty-two datasets aiming at detecting anomalies of smart home inhabitants' behaviour were created. Moreover, the anomalies are defined and annotated by the participants.

The work at hand conducted an evaluation of the current state of the HTM implementations. There are several implementations of the HTM theory and the most active implementation is NuPIC. The anomaly detection models in this implementation can use several encoders that encode the dataset records to be fed to the HTM model. An evaluation of the available encoders revealed the need to develop an encoder capable of dealing with datasets generated by a smart home environment. The smart home environment contains many sensors and this increases dimensionality of features set.

The HI-SDR encoder was developed as a novel encoder to overcome the issues discovered when using HTM models to detect anomalies in the smart home domain. The evaluation of the current state-of-the-art unsupervised anomaly detection algorithms and techniques on the forty-two smart home datasets revealed good and promising results. The HTM model with the HI-SDR encoder was able to achieve 81.9% accuracy using NAB as an anomaly detection metric with 17.8% increase over the k-NN algorithm and 47.5% increase over the standard CLA encoders.

7.1 Research Contributions

This work offers several contributions to knowledge and can be summarised as follows:

- The primary contribution of this research is the development of a novel HTM encoder (HI-SDR) to solve the issue of high-dimensional datasets in smart homes. The current implementation of HTM theory, NuPIC, offers several encoders that are suitable for scalar and categorical data types. However, these encoders are designed to work with a single feature (a single column) in the dataset or with few columns. When using several scalar or categorical encoders to encode each column, the resulting output will be big and less sparse than the HI-SDR encoder. Keeping the sparsity low for the output of the encoder is an important property as this will allow the HTM model to form meaningful representations to learn from. The smart home's environment usually consists of several sensors and actuators which affect the performance of the standard encoders. The HI-SDR encoder uses a hashing technique to reduce the dimensionality of the dataset feature space by encoding it into a representation that the rest of the HTM model can work with and produce good anomaly detection results.
- A novel smart home simulation tool, OpenSHS, targeted at researchers who are interested in smart home research area. OpenSHS represents a new hybrid, open-source, cross-platform, 3D smart home simulator aiming at dataset generation. The literature review of the current simulation tools had many shortcomings and OpenSHS is an attempt to rectify these issues.
- Forty-two datasets of smart home participant's daily activities with annotated anomalies. One of the issues revealed by the literature review of smart homes is the lack of a standardised dataset targeted at detecting anomalies in a smart home environment. Using OpenSHS, the participants simulated their daily habits and self-annotated these datasets. The collection of these datasets is available publicly online to allow for the whole research community to test and evaluate their machine learning models.

7.2 Conclusion and Future Work

Representing the input as SDRs is one of the main strengths and advantages of using an HTM based model. As long as the required properties for the encoders are met, HTM model can work with any type of data unlike some machine learning models. The HI-SDR encoder has been designed to work with high-dimensional binary datasets. However, not all the data in a smart home can be modelled into two states, on and off. Therefore, one of the main tasks for future work is to investigate how to integrate and fuse several data types into one SDR. Ideally, the encoder should be able to work with high-dimensional categorical and scalar data types. However, that would make the output of the encoder big and the challenge is to make the output size manageable without sacrificing needed information. Incorporating dimensionality reduction techniques such as PCA could provide insightful solutions.

The hierarchical arrangement of regions is one of the HTM theory main principals. However, the current implementation, NuPIC, does not have a hierarchy of regions and only implements one region. Therefore, the question is still open for how to create an efficient hierarchy of regions and how to pass the data between them. Moreover, once an efficient fusion and integration of hierarchies is implemented, the question will be, how to arrange these regions and what should each region learn. For example, in image and video recognition, the Convolutional Neural Network's (CNN) lower layers learn and recognise primitive shapes while the higher layers learn bigger shapes. Investigating how to arrange several HTM regions to detect anomalies where the lower regions learn smaller features and the higher regions learn more abstract patterns, will be an interesting challenge.

Another task for future work is to test and evaluate the performance of several full HTM models dedicated to each column in the dataset. The challenge in this setup is how to integrate their output and whether or not this approach will outperform the current setup.

The HTM models have several parameters that need to be tuned and optimised to get the best possible results. Particle Swarm Optimisation (PSO) was used to optimise the HTM model's parameters for classification problems. One of this research results can be used to provide a heuristic that can improve the selection of good parameters. The heuristic is the number of bursting columns in the TM. For future work, an investigation of whether it is possible to propose an optimisation technique based on this heuristic that could achieve better results in optimising HTM anomaly detection models.

Calculating the anomaly was done using statistical techniques. However, investigating biologically inspired techniques to calculate the anomaly score could improve the results even further.

The developed smart home simulation tool, OpenSHS, focused on streamlining the simulation process for the participants and the researchers. The flexibility, scalability and accessibility of the tool are an ongoing target for improvements. One of the future plans is to add a floor plan editor to make it easier for the researcher to bootstrap their smart home design. Adding more smart devices to the library of devices available with OpenSHS will help the flexibility and scalability of the tool. Since OpenSHS is open-source tool and publicly available for the research community, the library of simulated devices could see fast growth as new devices are introduced. As for the accessibility, porting OpenSHS to other platforms would lower the barriers for using it to conduct simulations. With the recent advancements in web technologies, OpenSHS could be ported to run in the web browser which will make the tool more accessible.

For future work, full multiple inhabitants support in real-time will be included. Moreover, the smart devices' library, has few specialised sensors that could be updated to include new types of sensors and devices. Taking into consideration that OpenSHS is an open-source project, released under a free and permissive licence, the project could envisage quick and rapid development that would facilitate the support of the aforementioned features.

The participants in this work used OpenSHS to generate forty-two datasets of their ADLs and patterns. The participants had total freedom to perform their activities and no restrictions were imposed by the researcher. The level of variability shown in the descriptive statistics presented in Section 6.2.2 reflect this fact. However, the main limitation for any simulation research is that it will not fully substitute an actual and real-world experiment.

The more realistic the simulation is, the less the need for building actual smart homes to carry out research. Following the growing advancements in computer graphics, Virtual Reality (VR) is becoming more accessible and affordable. BlenderVR (Katz *et al.*, 2015) is an open-source framework that extends Blender and allows it to produce immersive and realistic simulations. Since OpenSHS is based on Blender, one of our future goals is to investigate the incorporation of BlenderVR into our tool to provide more true to life experiences for the smart home simulation and visualisation. In terms of accessibility, the aim is to make OpenSHS as accessible as possible. Nowadays, the web technologies and web browsers can be a good platform to facilitate the wider distribution of OpenSHS. Technologies such as WebGL (2006) can be used to run OpenSHS in different web browsers and Blender can export to these technologies.

Currently, the labelling of activities is performed by the participant during the simulation phase. OpenSHS does not perform automatic recognition of these activities. As part of a future work, the possibility of adding automatic recognition of the participants' activities can be investigated.

The work presented in this thesis lends itself well to be implemented in a real smart home provided that a middleware layer is installed. The middleware is responsible for sampling, aggregating, and cleaning the data generated by the smart devices and the sensors. If the requirements explained in Section 1.3 are met, the work in this thesis should provide anomaly detection services to the middleware layer. However, more research in a real smart home setting is required as there are more potential for new problems emerging.

One of the important issues facing smart home research is the lack of representative and standardised datasets for anomaly detection tasks. It is crucial to have good datasets to validate any proposed anomaly detection technique. However, anomaly detection is a difficult task to undertake due to the nature of anomalies. Anomalies in a smart home scenario are rare events and have a subjective quality. Every smart home inhabitant has their own habits and patterns that makes it difficult to objectively decide whether an event is an anomaly or not. Due to this subjective quality, in this work the inhabitants themselves were the ones who decided if an event is anomaly according to their own habits. This was captured in the forty-two datasets that were generated by the participants. For future work, more datasets that follow the same methodology but explore more complex scenarios and simulate more than a single inhabitant are planned.

To evaluate the ability of an anomaly detection model to adapt to new normal patterns introduced by the inhabitants, more work is required to create datasets that capture complex and intricate patterns of the inhabitants. It is common that the normal patterns of the inhabitants will change over time due to changes in the inhabitants' life styles. Therefore, testing the adaptability of the anomaly detection model to cope with these changing patterns is important. For future work, more datasets are needed that incorporate several changing normal patterns of the inhabitants to test the ability of the anomaly detection model to recover and recognise the new normal patterns.

Appendix A

Spatial Pooler and Temporal Memory

In this appendix the pseudocode of the Spatial Pooler and the Temporal Memory will be presented based on the work of Hawkins *et al.* (2016).

A.1 Spatial Pooler

The Spatial Pooler receives the feedforward input coming from the encoders and learns the spatial feature of every input. The output of the Spatial Pooler is a set of active columns. After initialising the Spatial Pooler, the algorithm can be divided into three phases:

- Calculating the overlap between the columns and the input space.
- Calculating the winning columns after performing inhibition.
- Update the permanence of the synapses.

A.1.1 Initialisation

The first thing that occurs when creating a new Spatial Pooler instance is initialising the permanence values for the columns. Every column is assigned a random permanence value. This value is bounded by two conditions, all values should be normally distributed around the `connectedPerm` variable (A.1). The second condition is to have every column permanence biased towards the centre of the input space. Meaning, when overlaying the

input space over the active columns, higher permanence values are assigned to centre of every column and these values are gradually decreasing the further we go away from that centre.

A.1.2 First Phase: Overlap

Algorithm 3 Calculating the overlap between the input space and the Spatial Pooler active columns.

for c **in** $columns$ **do**
    OVERLAP$(c) \leftarrow 0$
    for s **in** CONNECTEDSYNAPSES(c) **do**
        OVERLAP$(c) \leftarrow$ OVERLAP$(c) +$ INPUT$(t, s.sourceInput)$
    end for

    if OVERLAP$(c) < minOverlap$ **then**
        OVERLAP$(c) \leftarrow 0$
    else
        OVERLAP$(c) \leftarrow$ OVERLAP$(c) \times$ BOOST(c)
    end if

end for

A.1.3 Second Phase: Inhibition

Algorithm 4 Calculating the winning columns after the inhibition.

for c **in** $columns$ **do**
    $minLocalActivity \leftarrow$ KTHSCORE(NEIGHBORS(c), $desiredLocalActivity$)
    if OVERLAP$(c) > 0$ **and** OVERLAP$(c) \geq minLocalActivity$ **then**
        ACTIVECOLUMNS(t).APPEND(c)
    end if
end for

A.1.4 Third Phase: Update

Algorithm 5 Updating the synapses permanence.

20: **for** c **in** ACTIVECOLUMNS(t) **do**
21:
22: **for** s **in** POTENTIALSYNAPSES(c) **do**
23: **if** ACTIVE(s) **then**
24: $s.permanence \mathrel{+}= permanenceInc$
25: $s.permanence \leftarrow$ MIN$(1.0,\ s.permanence)$
26: **else**
27: $s.permanence \mathrel{-}= permanenceDec$
28: $s.permanence \leftarrow$ MAX$(0.0,\ s.permanence)$
29: **end if**
30: **end for**
31: **end for**
32:
33: **for** c **in** $columns$ **do**
34: MINDUTYCYCLE$(c) \leftarrow 0.01 \times$ MAXDUTYCYCLE(NEIGHBORS(c))
35: ACTIVEDUTYCYCLE$(c) \leftarrow$ UPDATEACTIVEDUTYCYCLE(c)
36: BOOST$(c) \leftarrow$ BOOSTFUNCTION(ACTIVEDUTYCYCLE(c), MINDUTYCYCLE(c))
37:
38: OVERLAPDUTYCYCLE$(c) \leftarrow$ UPDATEACTIVEDUTYCYCLE(c)
39: **if** OVERLAPDUTYCYCLE$(c) <$ MINDUTYCYCLE(c) **then**
40: INCREASEPERMENANCES$(c,\ 0.1 \times connectedPerm)$
41: **end if**
42:
43: **end for**
44: $inhibitionRadius \leftarrow$ AVERAGERECEPTICEFIELDSIZE()

A.1.5 Functions and Data Structures

TABLE A.1: Spatial Pooler data structures.

Data Structure	Description
`activeColumns(t)`	A list of the indices of the winning columns.
`activeDutyCycle(c)`	An average number for how active column `c` was during a specified period.
`boost(c)`	The value for boosting column `c`.
`columns`	A list of all the columns.
`connectedPerm`	A threshold of a synapse that if exceeded, the synapse is said to be connected.
`connectedSynapses(c)`	A list of the connected synapses out of the potential synapses for column `c`.
`desiredLocalActivity`	The number of winning columns after the inhibition process.
`inhibitionRadius`	The average connected receptive field size.
`input(t, j)`	The input space items at time `t`. Its value is `1` if the j^{th} item bit is on.
`minDutyCycle(c)`	The minimum required activity of column `c`. If the column activity is below this threshold, it will be boosted. The threshold is 1% of the maximum activities of the column neighbors.
`minOverlap`	The minimum number of active input bits overlapping with the columns.
`neighbors(c)`	A list of all the columns falling into the `inhibitionRadius` of the column `c`.
`overlap(c)`	The overlap score of column `c` with the input space.
`overlapDutyCycle(c)`	An average number for how column `c` been passing the `minOverlap` threshold during a specified period.
`permanenceDec`	The amount by which to decrease the synapse.
`permanenceInc`	The amount by which to increase the synapse.
`potentialSynapses(c)`	A list of potential synapses and their performance for column `c`.
`synapse`	An object containing a permanence value and the index of the associated input bit.

TABLE A.3: Spatial Pooler functions and their returned values types.

Function	Type	Description
`kthScore(cls, k)`	`int`	Returns the k^{th} highest overlap score from a list of columns `cls`.
`updateActiveDutyCycle(c)`	`float`	Calculates an average of the activity of column `c` after inhibition during a specific period.
`updateOverlapDutyCycle(c)`	`float`	Calculates an average of how column `c` been passing the overlap threshold `minOverlap` during a specific period.
`averageRecepticeFieldSize()`	`float`	The average radius of the connected receptive field size for all columns.
`maxDutyCycle(cls)`	`float`	The maximum active duty cycle of the columns `cls`.
`increasePermenances(c, s)`	`void`	Increase the permanence of all synapses in column `c` by the factor `s`.
`boostFunction(c)`	`float`	Returns the boost factor of columns `c`. The boost value is ≥ 1. The value is 1 when the column activity is above the threshold `minDutyCycle`

A.2 Temporal Memory

A.2.1 Inference Mode

A.2.1.1 First Phase: Active Cells

Algorithm 6 Calculating the active cells in the Temporal Memory.

```
for c in ACTIVECOLUMNS(t) do

    buPredicted ← False
    for i ← 0 to cellsPerColumn − 1 do
        if PREDICTIVESTATE(c, i, t − 1) == True then
            s ← GETACTIVESEGMENT(c, i, t − 1, activeState)
            if s.sequenceSegment == True then
                buPredicted ← True
                ACTIVESTATE(c, i, t) ← 1
            end if
        end if

        if buPredicted == False then
            for i ← 0 to cellsPerColumn − 1 do
                ACTIVESTATE(c, i, t) ← 1
            end for
        end if
    end for
end for
```

A.2.1.2 Second Phase: Predictive Cells

Algorithm 7 Calculating the predictive cells in the Temporal Memory.

```
for c, i in cells do
    for s in SEGMENTS(c, i) do
        if SEGMENTACTIVE(c, i, s, t) then
            PREDICTIVESTATE(c, i, t) ← 1
        end if
    end for
end for
```

A.2.2 Learning Mode

A.2.2.1 First Phase: Active Cells

Algorithm 8 Calculating the active cells while learning in the Temporal Memory.

for c **in** ACTIVECOLUMNS(t) **do**
    $buPredicted \leftarrow$ **False**
    $lcChosen \leftarrow$ **False**
    for $i \leftarrow 0$ **to** $cellsPerColumn - 1$ **do**
        if PREDICTIVESTATE$(c,\ i,\ t-1) ==$ **True then**
            $s \leftarrow$ GETACTIVESEGMENT$(c,\ i,\ t-1,\ activeState)$
            if $s.sequenceSegment ==$ **True then**
                $buPredicted \leftarrow$ **True**
                ACTIVESTATE$(c,\ i,\ t) \leftarrow 1$
                if SEGMENTACTIVE$(s,\ t-1,\ learnState)$ **then**
                    $lcChosen \leftarrow$ **True**
                    LEARNSTATE$(c,\ i,\ t) \leftarrow 1$
                end if
            end if
        end if

        if $buPredicted ==$ **False then**
            for $i \leftarrow 0$ **to** $cellsPerColumn - 1$ **do**
                ACTIVESTATE$(c,\ i,\ t) \leftarrow 1$
            end for
        end if

        if $lcChosen ==$ **False then**
            $i,\ s \leftarrow$ GETBESTMATCHINGCELL$(c,\ t-1)$
            LEARNSTATE$(c,\ i,\ t) \leftarrow 1$
            $sUpdate \leftarrow$ GETSEGMENTACTIVESYNAPSES$(c,\ i,\ s,\ t-1,$ **True**$)$
            $sUpdate.sequenceSegment =$ **True**
            SEGMENTUPDATELIST.ADD$(sUpdate)$
        end if
    end for
end for

A.2.2.2 Second Phase: Predictive Cells

Algorithm 9 Calculating the predictive cells while learning in the Temporal Memory.

for $c,\ i$ **in** $cells$ **do**
  for s **in** SEGMENTS$(c,\ i)$ **do**
    if SEGMENTACTIVE$(s,\ t,\ activeState)$ **then**
      PREDICTIVESTATE$(c,\ i,\ t) \leftarrow 1$

      $activeUpdate \leftarrow$ GETSEGMENTACTIVESYNAPSES$(c,\ i,\ s,\ t, \mathbf{False})$
      SEGMENTUPDATELIST.ADD$(activeUpdate)$

      $predSegment \leftarrow$ GETBESTMATCHINGSEGMENT$(c,\ i,\ t-1)$
      $predUpdate \leftarrow$ GETSEGMENTACTIVESYNAPSES$(c,\ i,\ predSegment,\ t-1, \mathbf{True})$
      SEGMENTUPDATELIST.ADD$(predUpdate)$
    end if
  end for
end for

A.2.2.3 Third Phase: Update

Algorithm 10 Update the internal variables while learning in the Temporal Memory.

for $c,\ i$ **in** $cells$ **do**
  if LEARNSTATE$(s,\ i,\ t) == 1$ **then**
    ADAPTSEGMENTS(SEGMENTUPDATELIST$(c,\ i),\ \mathbf{True}$)
    SEGMENTUPDATELIST$(c,\ i)$.DELETE()
  else if PREDICTIVESTATE$(c,\ i,\ t) == 0$ **and** PREDICTIVESTATE$(c,\ i,\ t-1) == 1$ **then**
    ADAPTSEGMENTS(SEGMENTUPDATELIST$(c,\ i),\ \mathbf{False}$)
    SEGMENTUPDATELIST$(c,\ i)$.DELETE()
  end if
end for

A.2.3 Functions and Data Structures

TABLE A.5: Temporal Memory data structures.

Data Structure	Description
`activationThreshold`	The segment activation threshold. If the number of connected synapses in this segment is greater than this threshold, the segment is then active.
`activeColumns(t)`	A list of the indices of the winning columns at time `t`.
`activeState(c, i, t)`	A binary number that indicates that at column `c`, the i^{th} cell at time `t` is in an active.
`cell(c, i)`	The cells in column `c` and index `i`.
`cellsPerColumn`	The number of cells for each column.
`connectedPerm`	If a synapse permanence is greater than this threshold, the synapse is then connected.
`initialPerm`	The initial synapse permanence value.
`learnState(c, i, t)`	A binary number that indicates that if at column `c`, the i{th} cell is chosen to be the learning cell.
`learningRadius`	The area that a cell can form distal connections to.
`minThreshold`	The minimum segment activity for learning.
`newSynapseCount`	The maximum number of synapses to be added to a segment when learning.
`permanenceDec`	The value by which a synapse permanence is decreased during learning.
`permanenceInc`	The value by which a synapse permanence is increased during learning.
`predictiveState(c, i, t)`	A binary number that indicates that at column `c`, the i^{th} cell at time `t` is in a predictive state.
`segmentUpdate`	An object with three values: a segment index, a list of existing active synapses, and a boolean flag that indicates whether this segment should be the sequence segment.
`segmentUpdateList(c, i)`	A list of `segmentUpdate`'s for the i^{th} cell in column `c`.

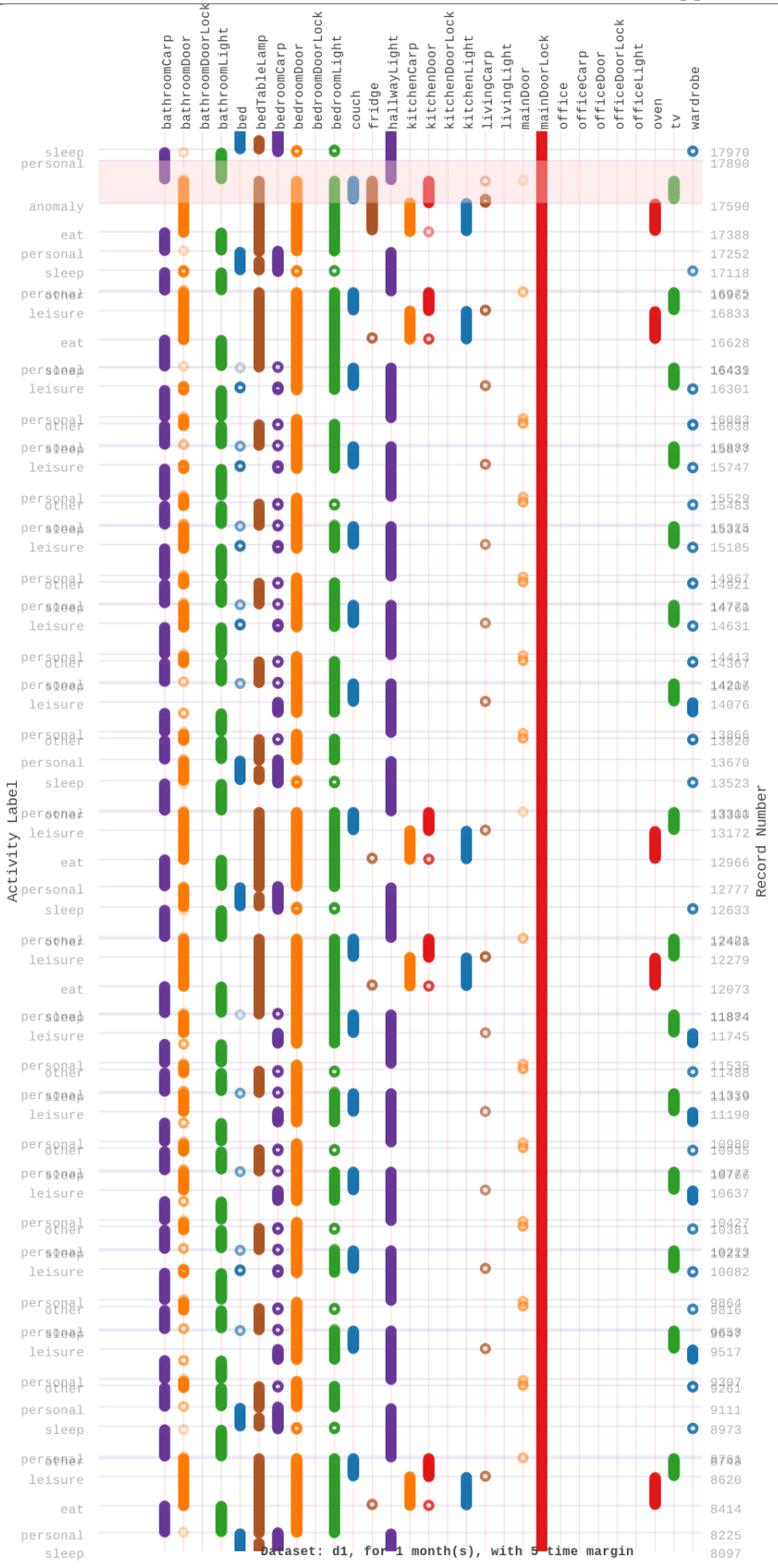
TABLE A.7: Temporal Memory functions and their returned values types.

Function	**Type**	**Description**
`segmentActive (s,t,state)`	`boolean`	Returns `True` if the number of connected synapses on the segment `s` that are activated by `state` at time `t` is greater than `activationThreshold`. `state` can be `activeState` or `learnState`.
`getActiveSegment (c,i,t,state)`	`int`	Returns a segment index for the i^{th} cell in column `c` at time `t` for which the segment `segmentActive(s, t, state)` is `True`. If multiple segments exists that satisfy the condition, sequence segments have precedence, then the segment with the highest activity
`getBest - MatchingSegment (c,i,t)`	`int`	Returns a segment index for the i^{th} cell in column `c` at time `t` for which the segment has the biggest number of active synapses. Returns -1 if no segments are found.
`getBest - MatchingCell(c)`	`int`	Returns a cell index for column `c` with the best matching segment. If no cell matches, it returns the index of the cell with the fewest segments.
`getSegment - ActiveSynapses (c,i,t,s,newSynapses)`	`segment - Update`	Returns a `segmentUpdate` object with changes to segment `s` for the i^{th} cell in column `c` at time `t`. The `newSynapses` is a optional boolean flag that defaults to `False`. If `newSynapses` is `True` then, `newSynapseCount` - count(activeSynapses) random learning synapses are added to `activeSynapses`.
`adaptSegments (segmentList, positive - Reinforcement)`	`void`	Goes through the list of `segmentUpdate` in the `segmentList` and if `positiveReinforcement` is `True`, then the active synapses permanence is increased by `permanenceInc`. Otherwise, the synapses permanence is decreased by `permanenceDec`.

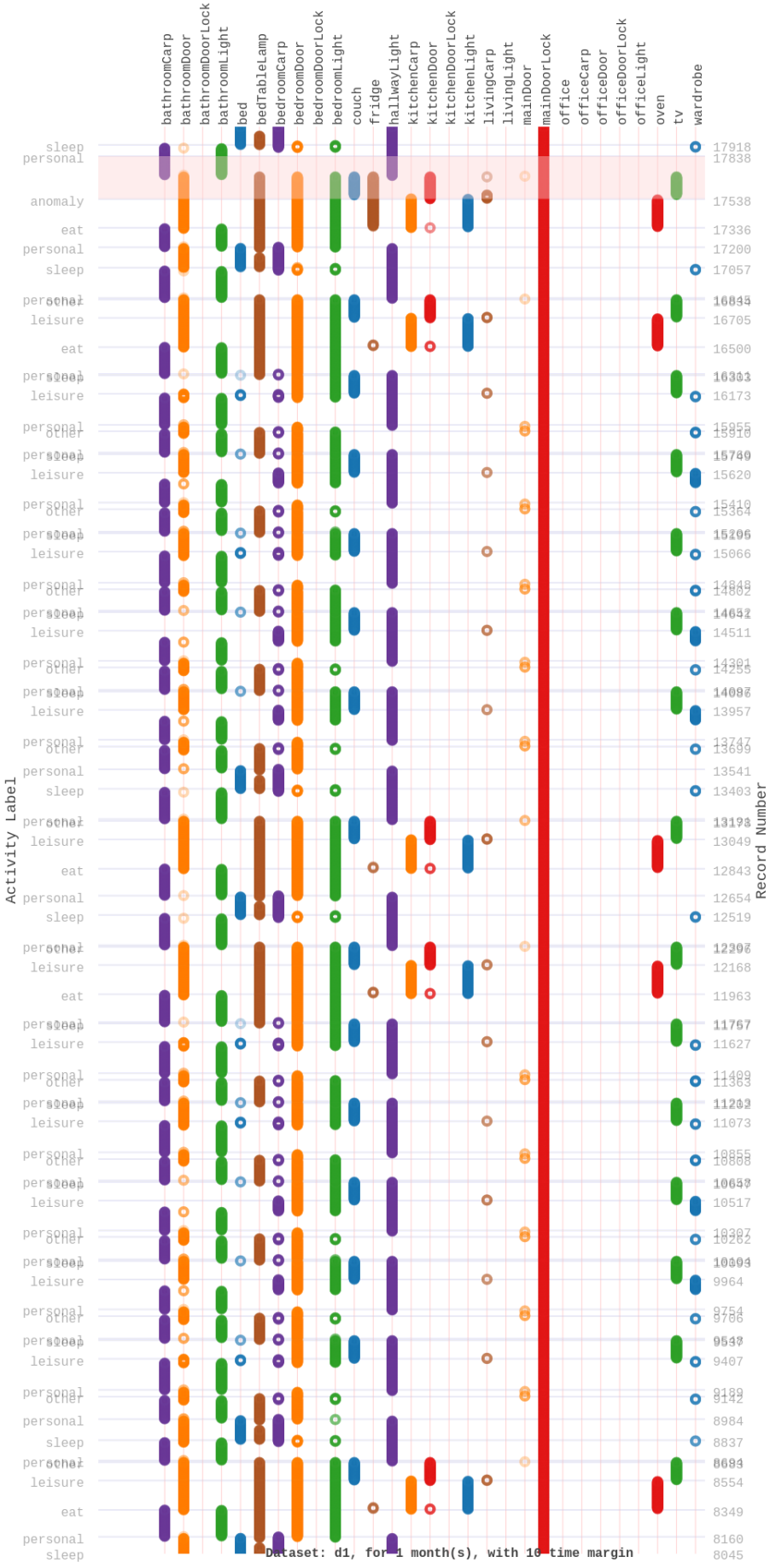
Appendix B

Datasets

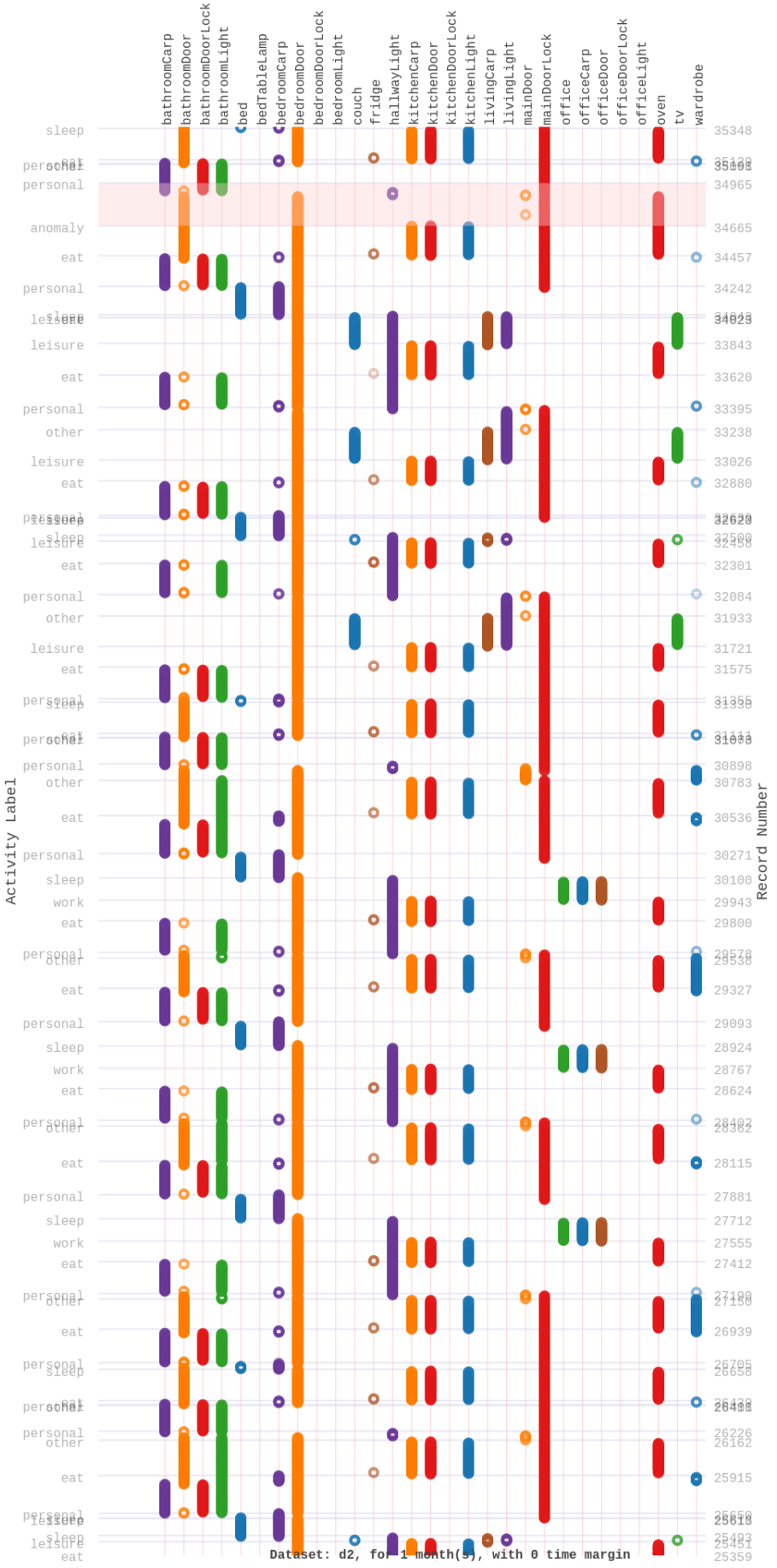
The last 10000 records of dataset d1-1m-0tm



The last 10000 records of dataset d1-1m-5tm

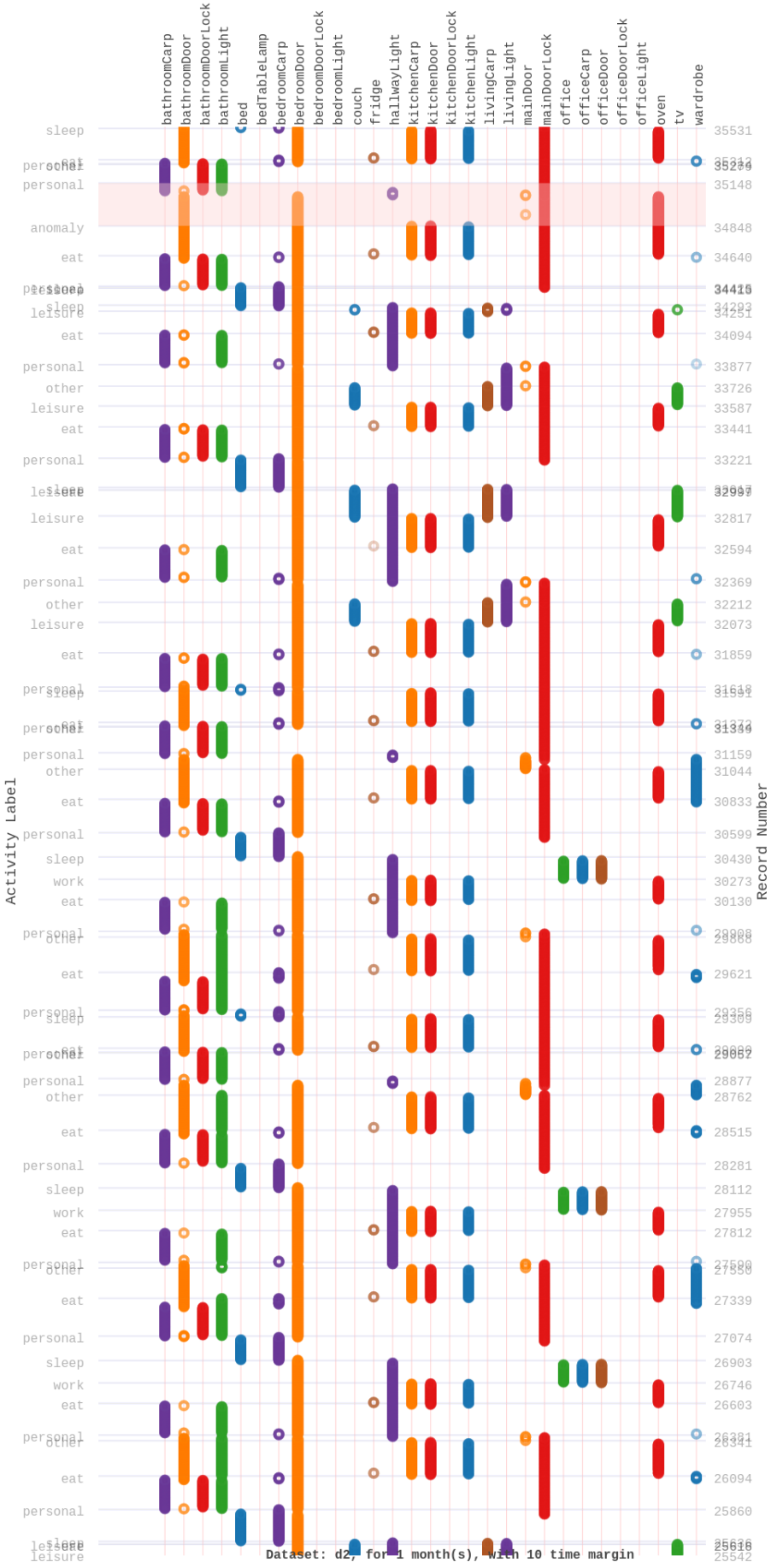


The last 10000 records of dataset d1-1m-10tm

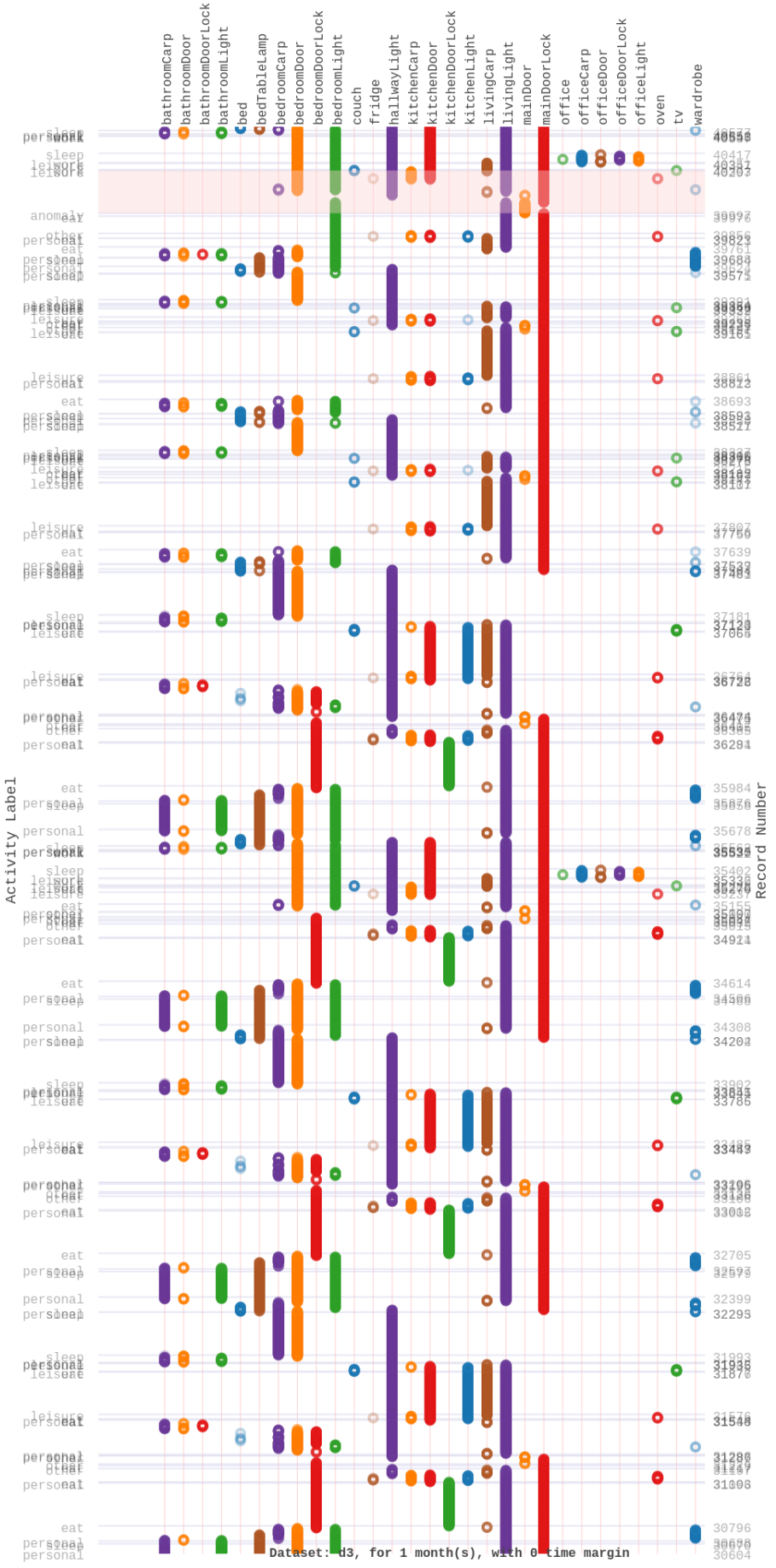


The last 10000 records of dataset d2-1m-0tm

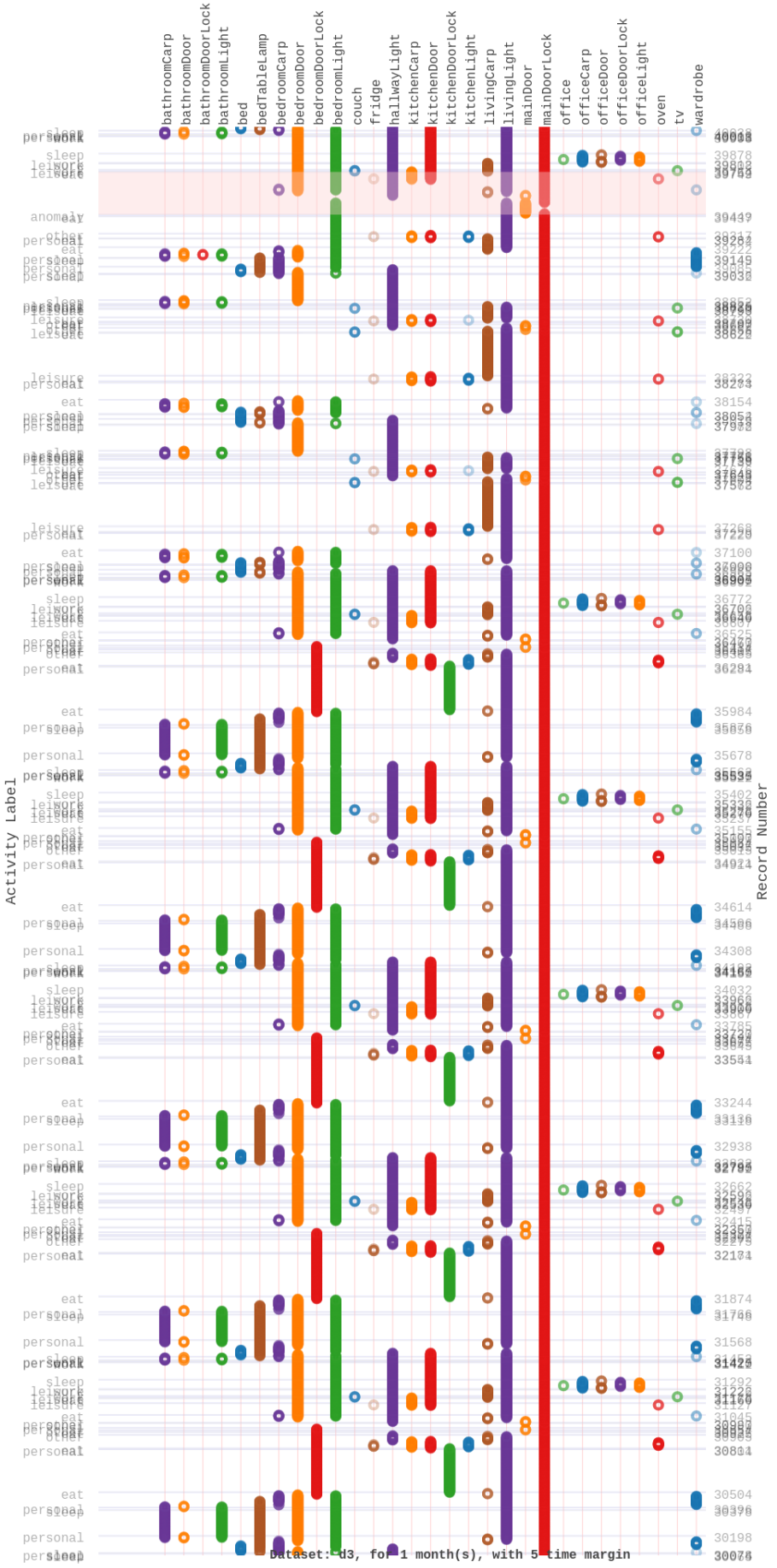
The last 10000 records of dataset d2-1m-5tm



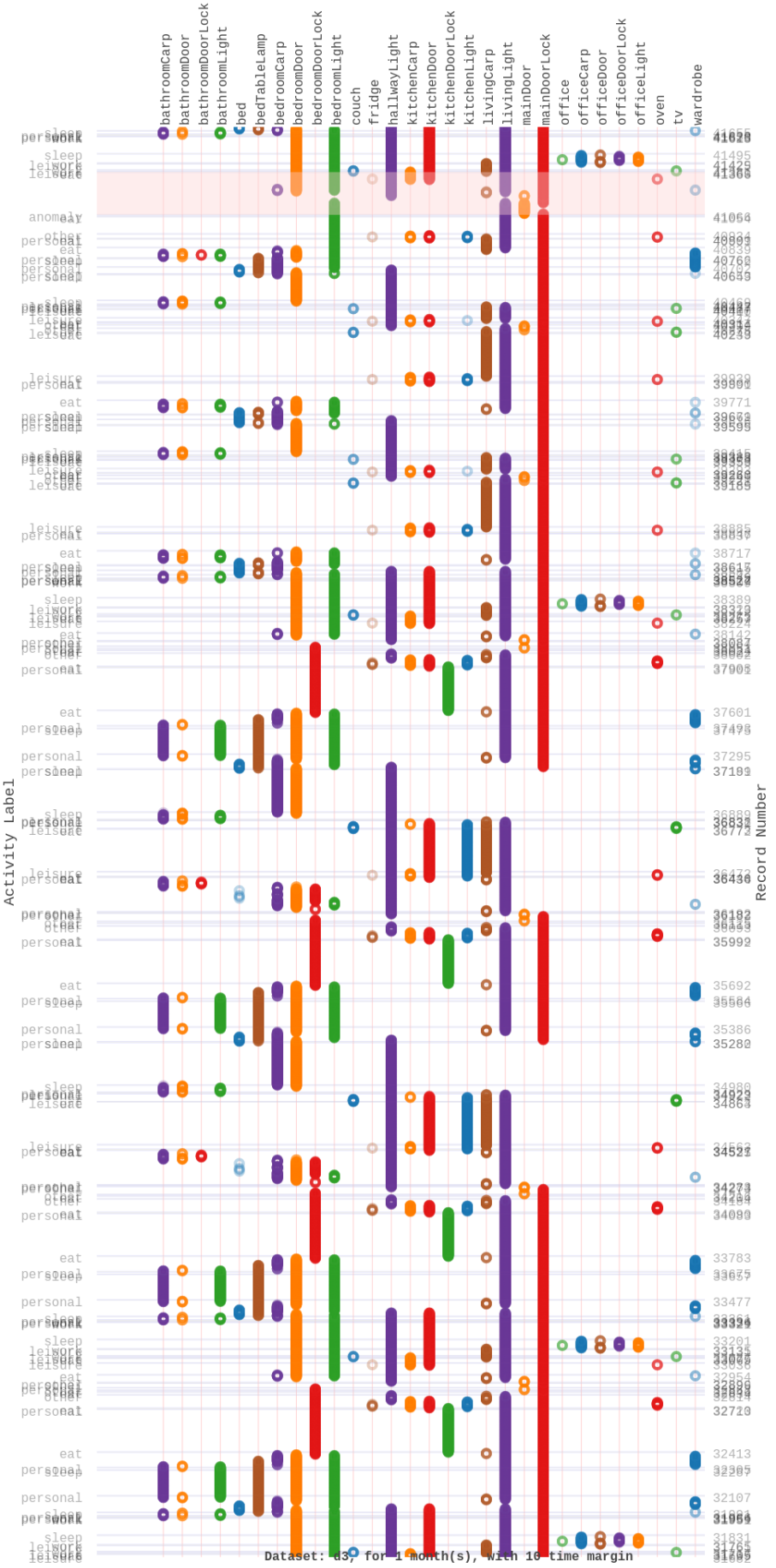
The last 10000 records of dataset d2-1m-10tm



The last 10000 records of dataset d3-1m-0tm

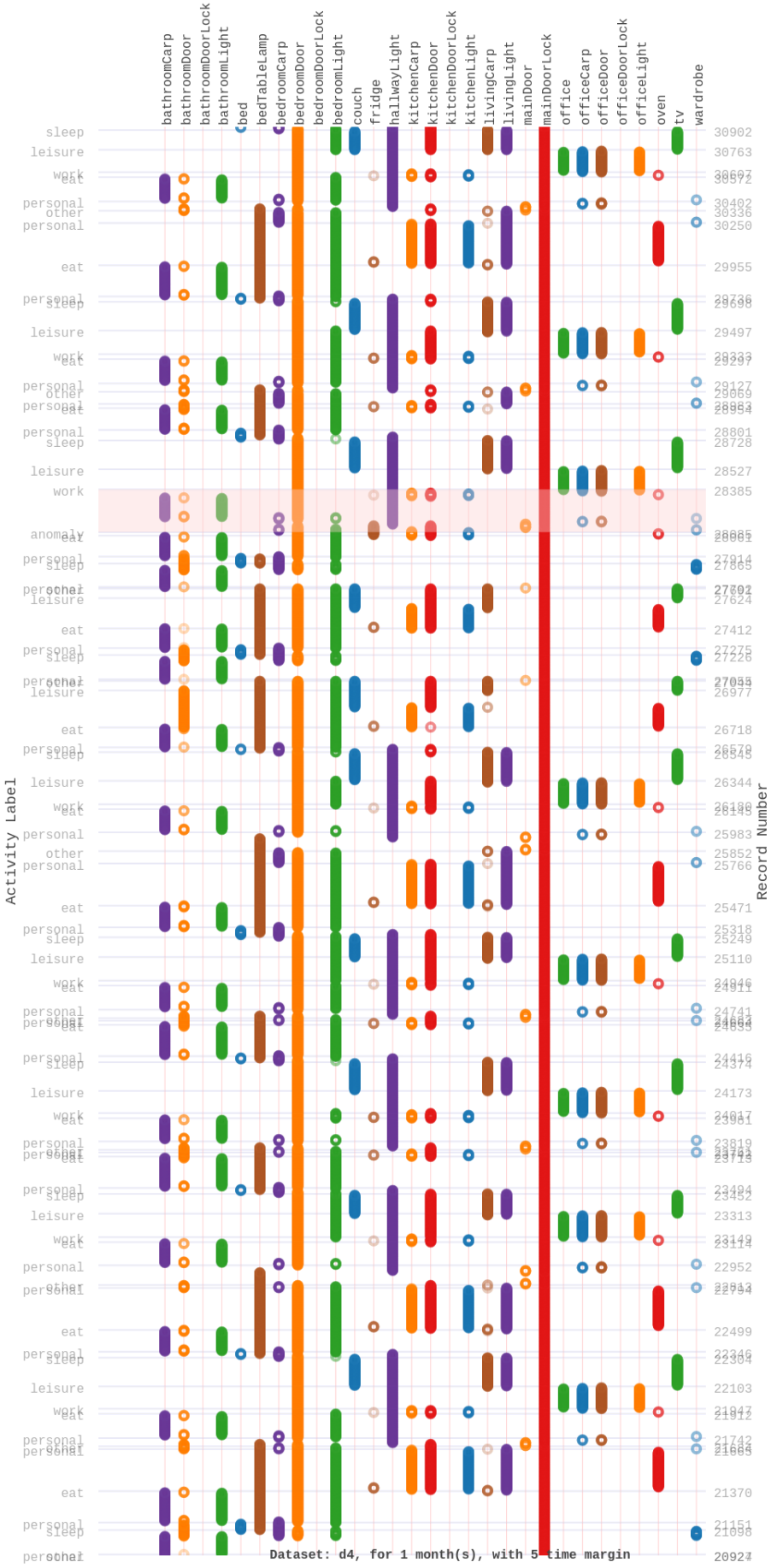


The last 10000 records of dataset d3-1m-5tm

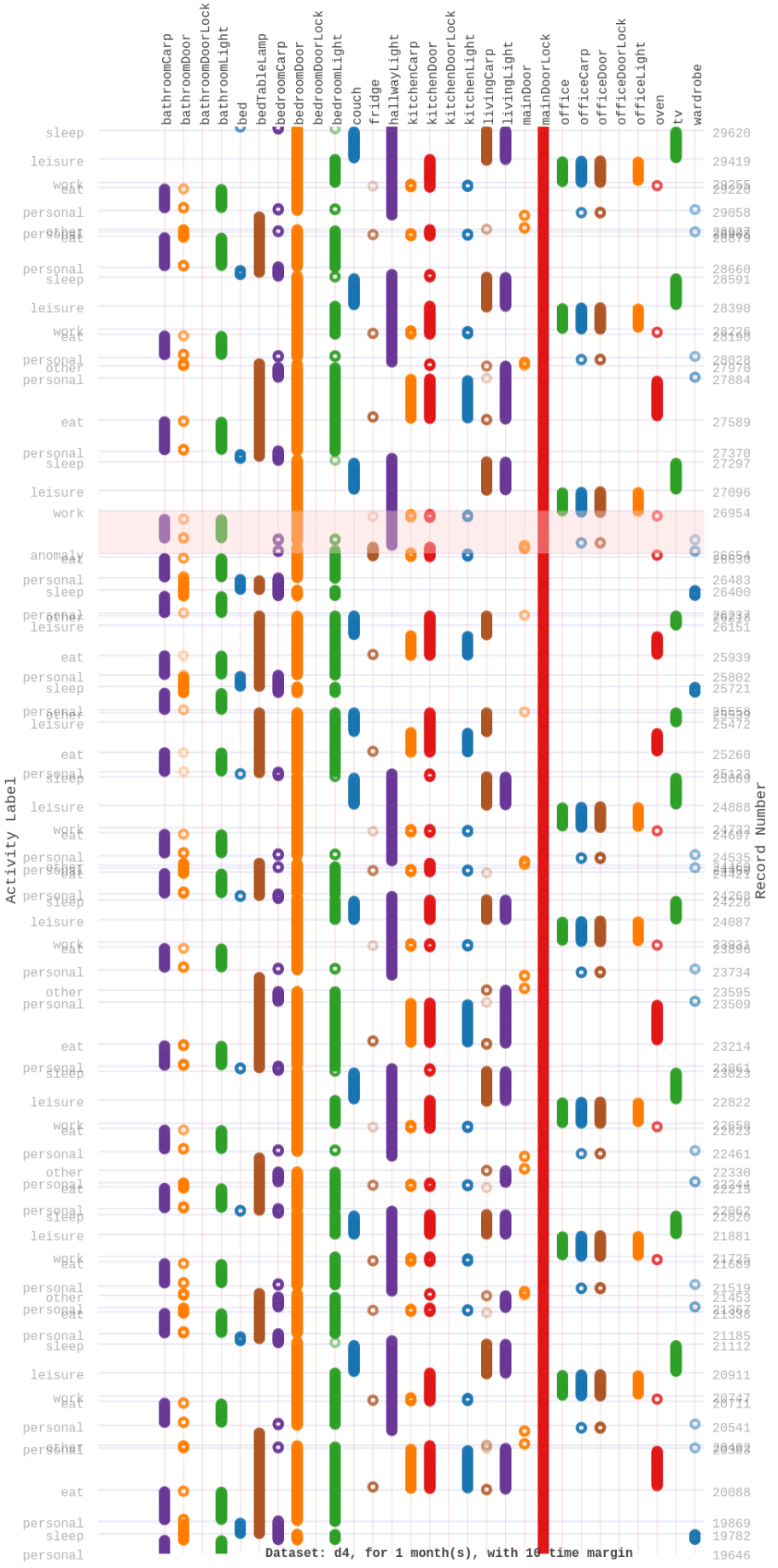


The last 10000 records of dataset d3-1m-10tm

The last 10000 records of dataset d4-1m-0tm

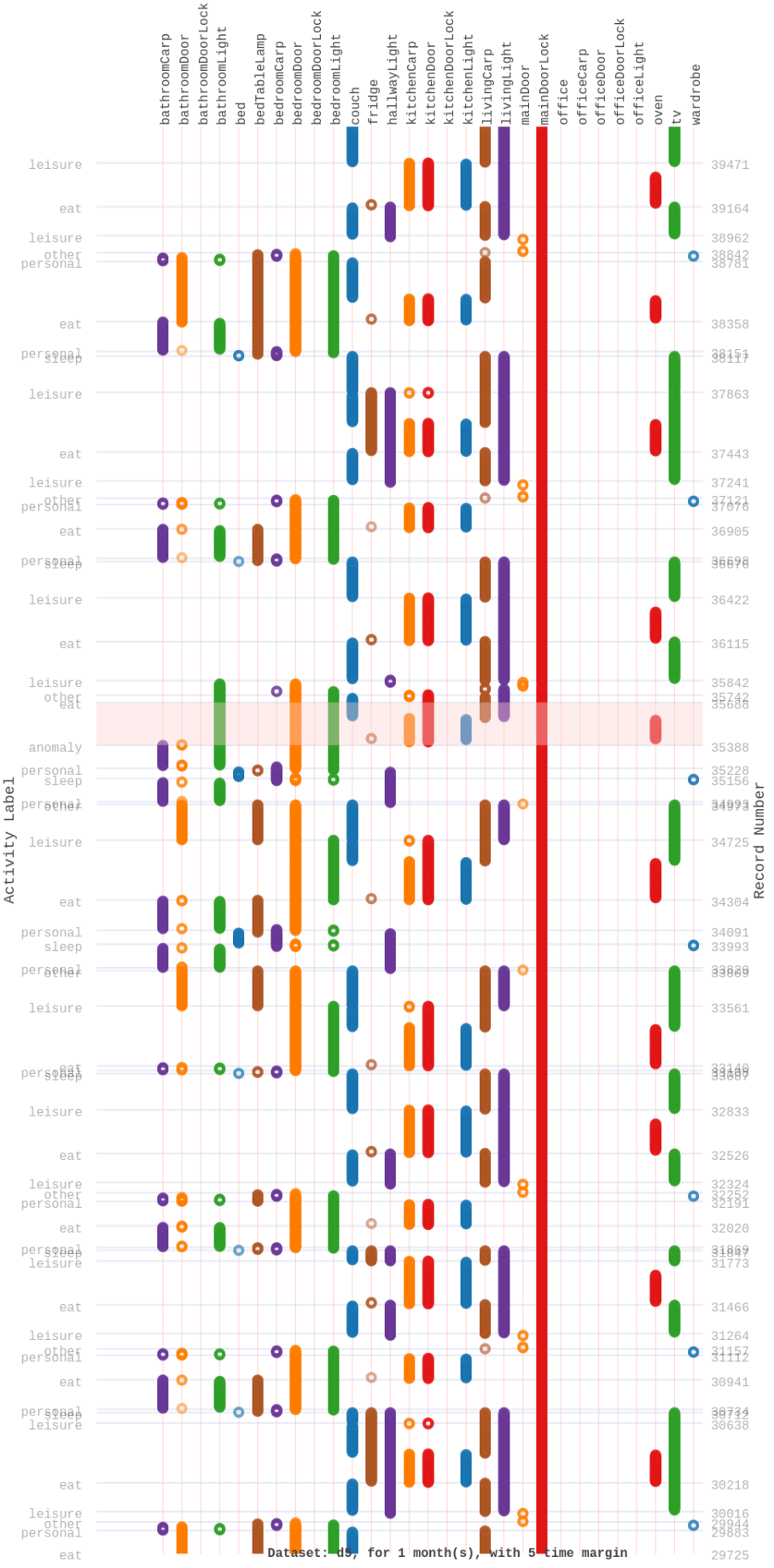


The last 10000 records of dataset d4-1m-5tm

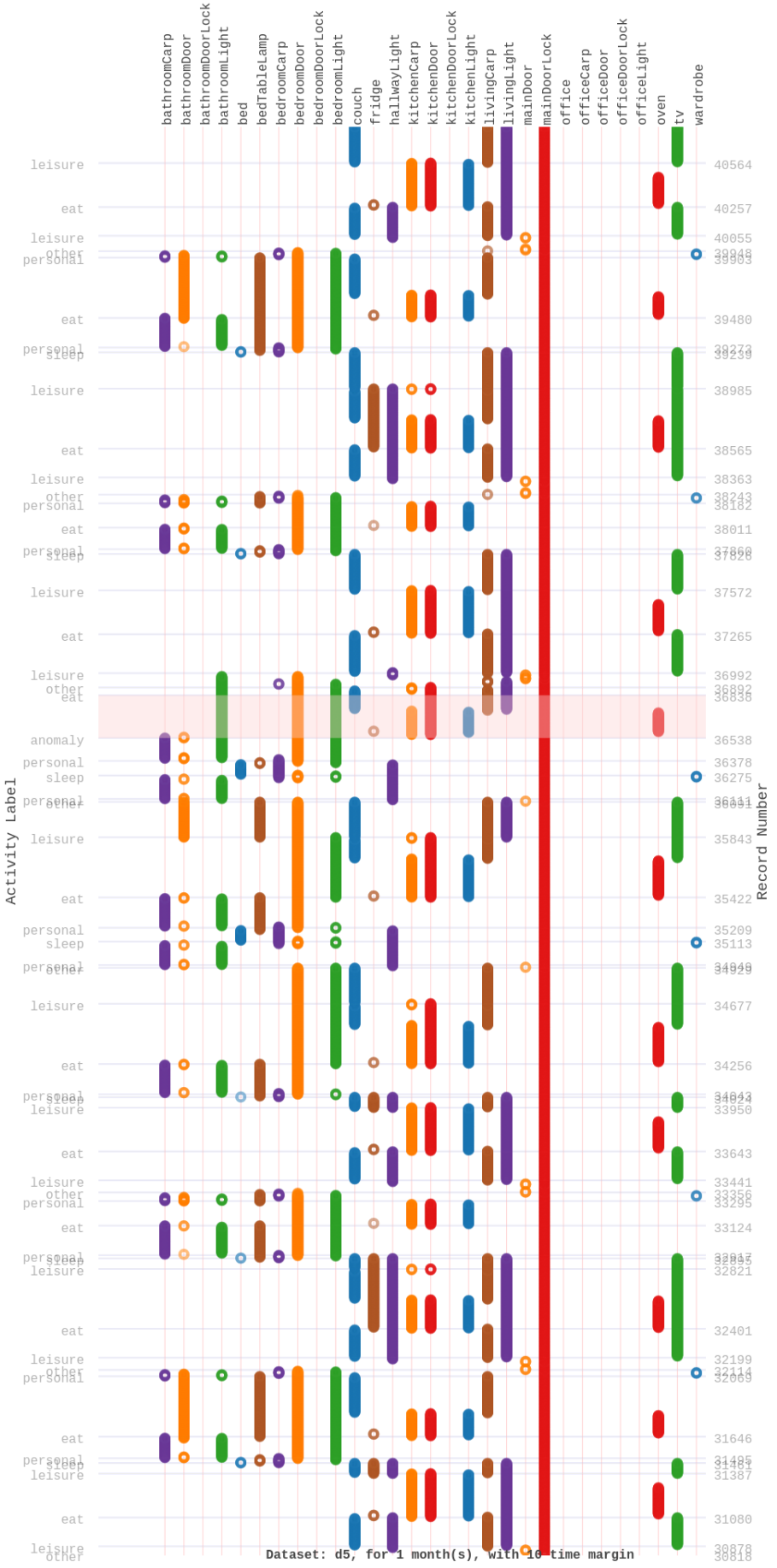


The last 10000 records of dataset d4-1m-10tm

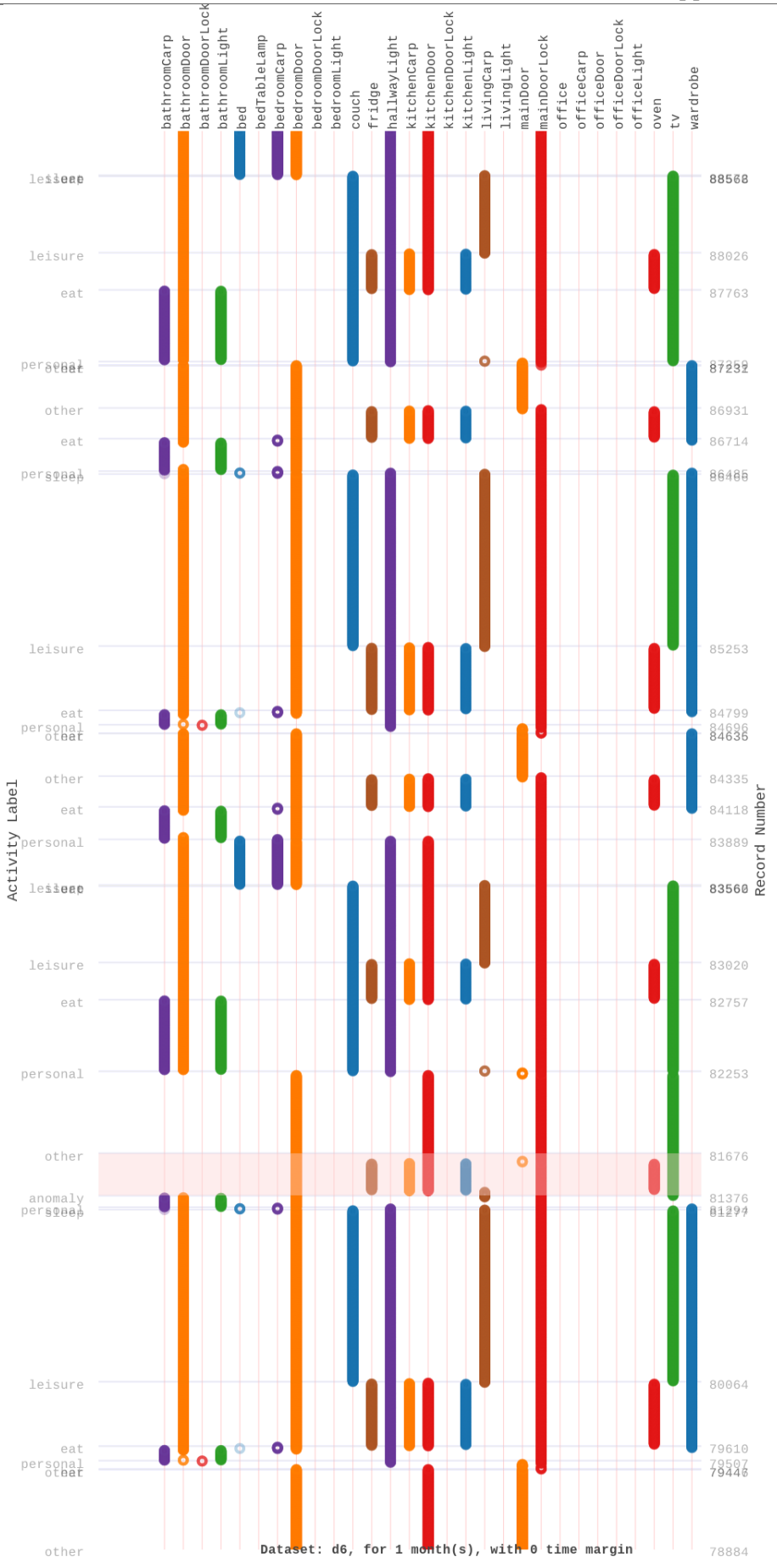
The last 10000 records of dataset d5-1m-0tm



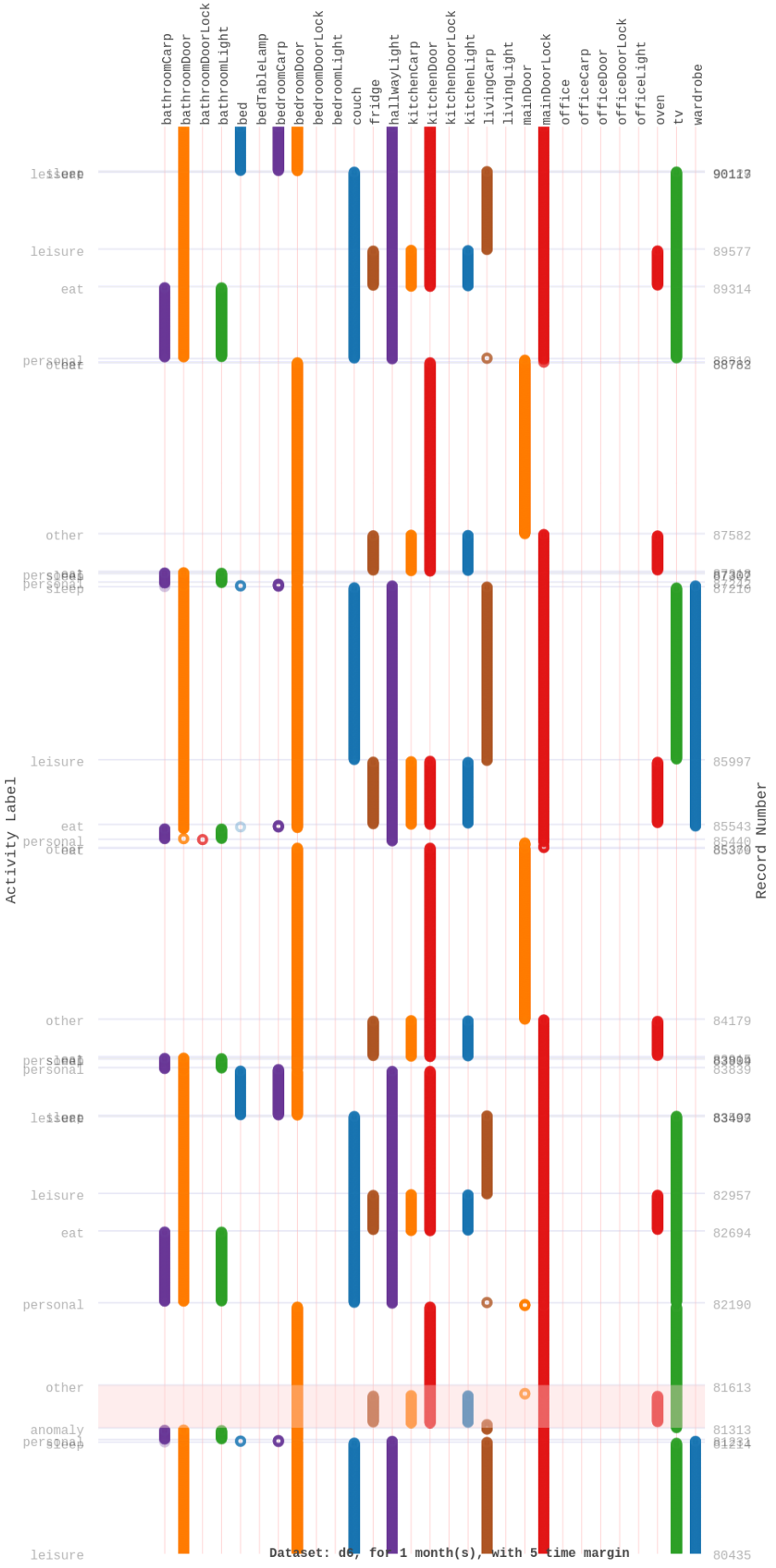
The last 10000 records of dataset d5-1m-5tm



The last 10000 records of dataset d5-1m-10tm

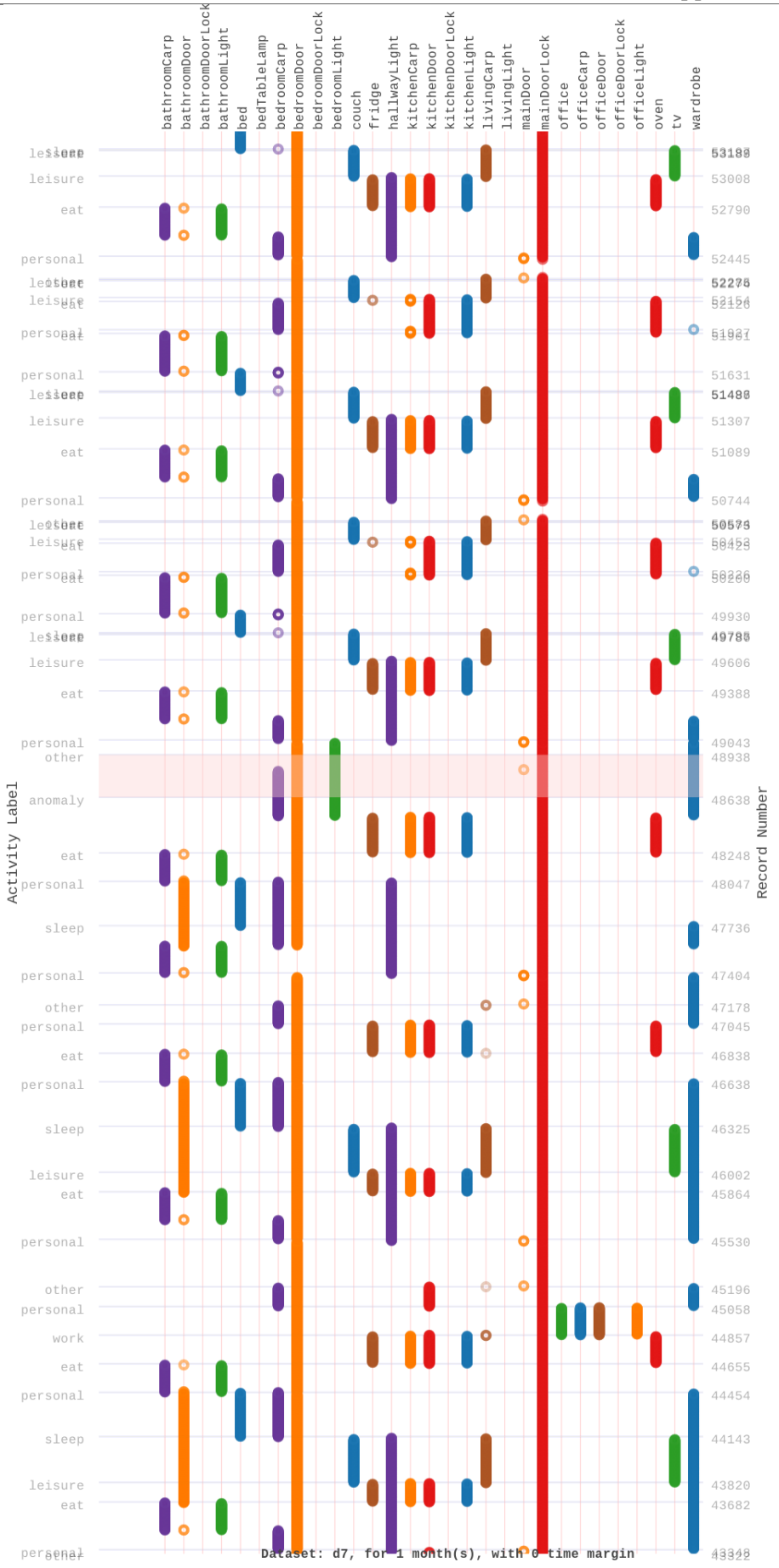


The last 10000 records of dataset d6-1m-0tm

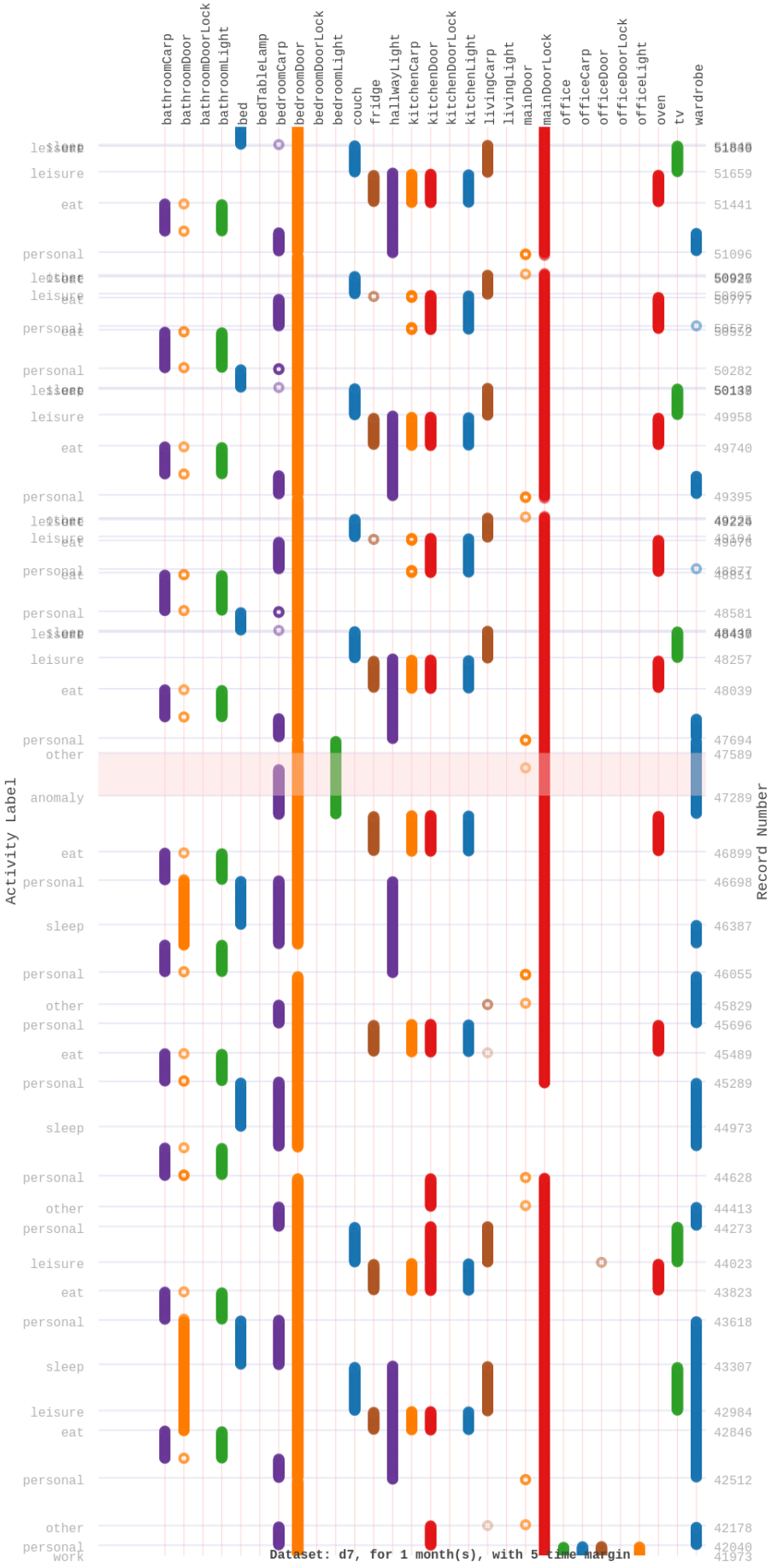


The last 10000 records of dataset d6-1m-5tm

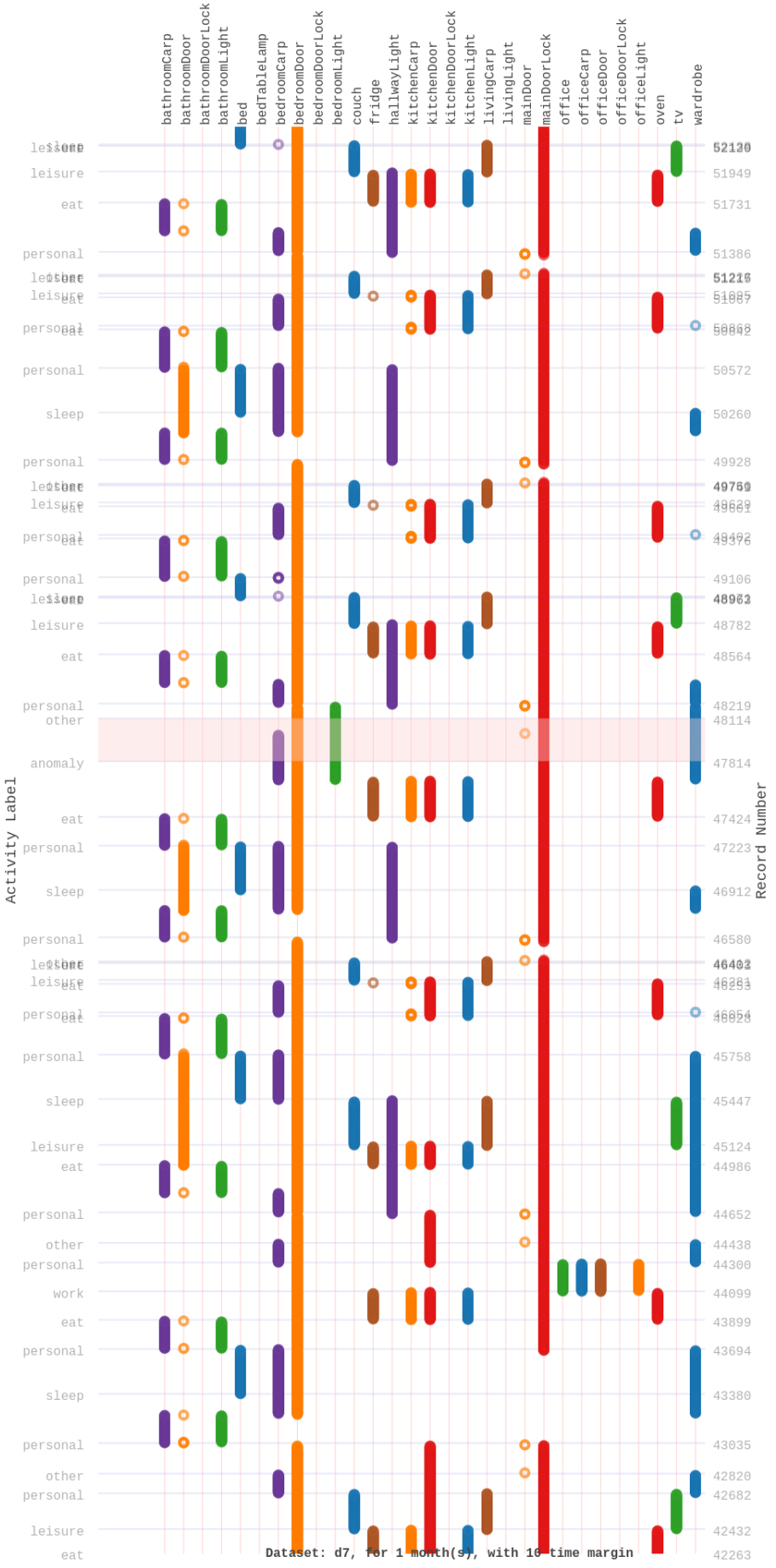
The last 10000 records of dataset d6-1m-10tm



The last 10000 records of dataset d7-1m-0tm



The last 10000 records of dataset d7-1m-5tm



The last 10000 records of dataset d7-1m-10tm

Appendix C

OpenSHS Documentation

C.1 Requirements

This section lists the required dependencies and their corresponding versions for OpenSHS as follows:

- **Blender**: version 2.74 or newer,
- **Python**: version 3.5 or newer,
- **Click**: version 6.6 or newer.

C.2 Quick Start

To start the simulation demo, from the root directory of OpenSHS, run the following commands:

```
cd app/
python openshs start -c morning
```

This will starts a blender session with the morning context simulation. Start the simulation by clicking <p> on the keyboard. All the interactoins will be captured and saved into the directory <app/temp>.

After doing mulitple simulations for each context (weekday morning, weekday evenings, weekend morning, weekend evenings), aggregate the final dataset by running the following command:

```
python openshs aggregate -d 30 -sd 2016-02-01 -tm 10
```

This will generate 30 days worth of data starting from 2016-02-01 and with a time margin of 10 minutes. The final dataset will be placed in the directory `<app/datasets>`.

C.3 Manual

This section describes all the currently available command line options for OpenSHS. There are three commands available for OpenSHS, `status`, `start`, and `aggregate`. Each one of these commands has its own options.

C.3.1 Start Command

The `start` command starts OpenSHS 3D environment with the specified context. This command has one option `--context` which specifies the context that should be started. The `--context` can be abbreviated as `-c`.

C.3.1.1 Examples

To start the `morning` context, for example:

```
python openshs start --context morning
```

C.3.2 Status Command

The `status` shows the status of the simulation session. There are two options for this command and one of them must be provided. The options are:

- `--list-contexts`: Lists the available contexts in this simulations. Can be abbreviated as `-lc`.
- `--recorded-samples`: Shows the status of the recorded contexts samples. Can be abbreviated as `-rs`.

C.3.2.1 Examples

To see a list of the available contexts for this simulation:

```
python openshs status --list-contexts
```

Which will output something similar to this:

```
morning, evening
```

To see a list of the recorded samples:

```
python openshs status -rs
```

Which will output something similar to this:

```
For context morning, weekdays: 3 Samples.
For context morning, weekends: 2 Samples.
For context evening, weekdays: 3 Samples.
For context evening, weekends: 2 Samples.
```

C.3.3 Aggregate Command

The `aggregate` command generates the final dataset from the recorded samples and store the as `<datasets/dataset.csv>`. This command has four options, as follows:

- `--days`: Specifies how many days to be generated. Can be abbreviated as `-d`.
- `--start-date`: Specifies the starting date of the dataset. In other words, the first date for the first record in the dataset. Can be abbreviated as `-sd`.
- `--time-margin`: The starting time margin for each replicated sample. More information on this option are described in Section 4.3.3.1. Can be abbreviated as `-tm`.
- `--variable-activitie`: Make the activities duration variable. Can be abbreviated as `-va`.

C.3.3.1 Examples

To aggregate the final dataset, a command similar to the following can be executed:

```
python openshs aggregate -d 30 -sd 2017-02-01 -tm 10 -va
```

This will aggregate all the recorded samples and generate a month worth of data (30 days) starting from 2017-02-01, with a time margin of 10 minutes and with variable activity lengths. The final dataset will be saved to the directory `<app/datasets/>`

References

Abraham, B. & Box, G. E. (1979). Bayesian analysis of some outlier problems in time series. *Biometrika*, (pp. 229–236).

Abraham, B. & Chuang, A. (1989). Outlier detection and time series modeling. *Technometrics*, 31(2), 241–248.

Agarwal, D. (2005). An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *Data Mining, Fifth IEEE International Conference on* (pp. 8–pp).: IEEE.

Agarwal, D. (2007). Detecting anomalies in cross-classified streams: a bayesian approach. *Knowledge and information systems*, 11(1), 29–44.

Aggarwal, C. C. (2013). *Outlier Analysis - Hull.PDF*. Springer Science & Business Media.

Aggarwal, C. C. & Yu, P. S. (2001). Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01 (pp. 37–46). New York, NY, USA: ACM.

Aggarwal, C. C. & Yu, P. S. (2008). Outlier detection with uncertain data. In *Proceedings of the 2008 SIAM International Conference on Data Mining* (pp. 483–493).: SIAM.

Agovic, A., Banerjee, A., Ganguly, A. R., & Protopopescu, V. (2008). Anomaly detection in transportation corridors using manifold embedding. *Knowledge Discovery from Sensor Data*, (pp. 81–105).

Ahmad, S. & Hawkins, J. (2015). Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory. *ArXiv e-prints*.

Ahmad, S. & Hawkins, J. (2016). How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. *ArXiv e-prints*.

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*.

Aitenbichler, E., Kangasharju, J., & Mühlhäuser, M. (2007). MundoCore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4), 332–361.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4), 393–422.

Alam, M. R., Reaz, M. B. I., & Ali, M. A. M. (2012). A review of smart homes—past, 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. (2013). 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.

Aleskerov, E., Freisleben, B., & Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. In *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997* (pp. 220–226).: IEEE.

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

Alshammari, N., Alshammari, T., Sedky, M., Champion, J., & Bauer, C. (2017b). openshs/openshs: First alpha release. `https://doi.org/10.5281/zenodo.274214`.

Alshammari, N. O., Mylonas, A., Sedky, M., Champion, J., & Bauer, C. (2015). Exploring the adoption of physical security controls in smartphones. In *International Conference on Human Aspects of Information Security, Privacy, and Trust* (pp. 287–298).: Springer International Publishing.

Amer, M. & Goldstein, M. (2012). Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)* (pp. 1–12).

Anderson, D., Frivold, T., & Valdes, A. (1995). Next-generation intrusion detection expert system (nides): A summary.

Angiulli, F. & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 15–27).: Springer.

Anscombe, F. J. (1960). Rejection of outliers. *Technometrics*, 2(2), 123–146.

Antic, S. D., Zhou, W.-L., Moore, A. R., Short, S. M., & Ikonomu, K. D. (2010). The decade of the dendritic nmda spike. *Journal of neuroscience research*, 88(14), 2991–3001.

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.

Armac, I. & Retkowitz, D. (2007). Simulation of smart environments. In *IEEE International Conference on Pervasive Services* (pp. 257–266).: IEEE.

Ashton, K. (2009). That "internet of things" thing. *RFiD Journal*, 22, 97–114.

Atallah, M., Szpankowski, W., & Gwadera, R. (2004). Detection of significant sets of episodes in event sequences. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (pp. 3–10).: IEEE.

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805.

Atzori, L., Iera, A., & Morabito, G. (2014). From "Smart Objects" to "Social Objects": The Next Evolutionary Step of the Internet of Things. *Communications Magazine, IEEE*, 52(January), 97–105.

Atzori, L., Iera, A., Morabito, G., & Nitti, M. (2012). The Social Internet of Things (SIoT) - When social networks meet the Internet of Things: Concept, architecture and network characterization. *Computer Networks*, 56(16), 3594–3608.

Augusteijn, M. & Folkert, B. (2002). Neural network classification and novelty detection. *International Journal of Remote Sensing*, 23(14), 2891–2902.

Auto-ID Labs (2003). Auto-id labs. `https://autoidlabs.org/`. (accessed on 15 February 2016).

Automation, R. (2000). Arena simulation software. `http://www.arenasimulation.com/`. (accessed on 20 December 2016).

Baker, L. D., Hofmann, T., McCallum, A., & Yang, Y. (1999). A hierarchical probabilistic model for novelty detection in text. In *Proceedings of International Conference on Machine Learning*.

Ballagas, R., Szybalski, A., & Fox, A. (2004). Patch panel: enabling control-flow interoperability in ubicomp environments. *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, 2004.*, (pp. 241–252).

Bandyopadhyay, D. & Sen, J. (2011). Internet of Things: Applications and Challenges in Technology and Standardization. *Wireless Personal Communications*, 58(1), 49–69.

Barbara, D., Wu, N., & Jajodia, S. (2001). Detecting novel network intrusions using bayes estimators. In *Proceedings of the 2001 SIAM International Conference on Data Mining* (pp. 1–17).: SIAM.

Barnet, V. (1976). The ordering of multivariate data (with discussion). *Journal of the Royal Statistics Society, Series A*, 139, 318–354.

Barnett, V. & Lewis, T. (1964). *Outliers in statistical data*. Chichester: John Wiley, 1995. 584p.

Barton, J. J. & Vijayaraghavan, V. (2002). Ubiwise, a ubiquitous wireless infrastructure simulation environment. *HP Labs*.

Beckman, R. J. & Cook, R. D. (1983). Outlier. s. *Technometrics*, 25(2), 119–149.

Bellman, R. E. (2015). *Adaptive control processes: a guided tour*. Princeton university press.

Bianco, A. M., Garcia Ben, M., Martinez, E., & Yohai, V. J. (2001). Outlier detection in regression models with arima errors using robust estimates. *Journal of Forecasting*, 20(8), 565–579.

Bishop, C. M. (1994). Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4), 217–222.

Blender (1995). Blender. `https://www.blender.org`. (accessed on 06 November 2016).

Bolton, R. J., Hand, D. J., *et al.* (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, (pp. 235–255).

Bormann, C., Castellani, A., & Shelby, Z. (2012). CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*, 16(2), 62–67.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144–152).: ACM.

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.

Box, G. E. & Tiao, G. C. (1968). A bayesian approach to some outlier problems. *Biometrika*, (pp. 119–129).

Branch, J. W., Giannella, C., Szymanski, B., Wolff, R., & Kargupta, H. (2013). In-network outlier detection in wireless sensor networks. *Knowledge and information systems*, 34(1), 23–54.

Brause, R., Langsdorf, T., & Hepp, M. (1999). Neural data mining for credit card fraud detection. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on* (pp. 103–106).: IEEE.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2), 93–104.

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

Brock, D. (2001). The Electronic Product Code (EPC): A Naming Scheme for Physical Objects. a note.

Brockett, P. L., Xia, X., & Derrig, R. A. (1998). Using kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. *Journal of Risk and Insurance*, (pp. 245–274).

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

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.

Byers, S. & Raftery, A. E. (1998). Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442), 577–584.

Byrne, F. (2015). *Real Machine Intelligence with Clortex and NuPIC*. Leanpub.

Cabrera, J. B., Lewis, L., & Mehra, R. K. (2001). Detection and classification of intrusions and faults using sequences of system calls. *Acm sigmod record*, 30(4), 25–34.

Campbell, M., Hoane, A. J., & Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1-2), 57–83.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., & Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4), 891–927.

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

Chan, P. K., Mahoney, M. V., & Arshad, M. H. (2003). A machine learning approach to anomaly detection. *Department of Computer Sciences, Florida Institute of Technology, Melbourne*.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41, 1–58.

Chandola, V., Boriah, S., & Kumar, V. (2008). Understanding categorical similarity measures for outlier detection. *Technical report 08–008, University of Minnesota*.

Chaqfeh, M. & Mohamed, N. (2012). Challenges in middleware solutions for the internet of things. In *2012 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 21–26).

Chatzigiannakis, V., Papavassiliou, S., Grammatikou, M., & Maglaris, B. (2006). Hierarchical anomaly detection in distributed large-scale sensor networks. In *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on* (pp. 761–767).: IEEE.

Chaudhary, A., Szalay, A. S., & Moore, A. W. (2002). Very fast outlier detection in large multidimensional data sets. In *DMKD*: Citeseer.

Chen, D., Shao, X., Hu, B., & Su, Q. (2005a). Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra. *Analytical Sciences*, 21(2), 161–166.

Chen, G., Branch, J., Pflug, M., Zhu, L., & Szymanski, B. (2005b). Sense: a wireless sensor network simulator. In *Advances in pervasive computing and networking* (pp. 249–267). Springer.

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

Chklovskii, D. B., Mel, B., & Svoboda, K. (2004). Cortical rewiring and information storage. *Nature*, 431(7010), 782–788.

Cichon, J. & Gan, W.-B. (2015). Branch-specific dendritic ca2+ spikes cause persistent synaptic plasticity. *Nature*, 520(7546), 180–185.

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

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

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., Youngblood, G. M., Heierman III, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. (2003b). Mavhome: An agent-based smart home. In *PerCom*, volume 3 (pp. 521–524).

Crook, P. & Hayes, G. (2001). A robot implementation of a biologically inspired method for novelty detection. In *Proceedings of Towards Intelligent Mobile Robots Conference*.

Crook, P. A., Marsland, S., Hayes, G., & Nehmzow, U. (2002). A tale of two filters-online novelty detection. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4 (pp. 3894–3899).: IEEE.

Da, C., Xueguang, S., Bin, H., & Qingde, S. (2005). Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra. *Analytical Sciences*, 21(2), 161–166.

Dasgupta, D. & Nino, F. (2000). A comparison of negative and positive selection algorithms in novel pattern detection. In *Systems, man, and cybernetics, 2000 IEEE international conference on*, volume 1 (pp. 125–130).: IEEE.

Davy, M. & Godsill, S. (2002). Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 2 (pp. II–1313).: IEEE.

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.

De Sousa Webber, F. (2015). Semantic Folding Theory And its Application in Semantic Fingerprinting. *ArXiv e-prints*.

De Stefano, C., Sansone, C., & Vento, M. (2000). To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1), 84–94.

De Villiers, M. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. In *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (pp. 142–151).: South African Institute for Computer Scientists and Information Technologists.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*, SE-13(2), 222–232.

Desforges, M., Jacob, P., & Cooper, J. (1998). Applications of probability density estimation to the detection of abnormal conditions in engineering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 212(8), 687–703.

Devineni, A. (2015). How our brains learn. `http://www.brains-explained.com/how-our-brains-learn`. (accessed on 11 October 2016).

Dey, A., Abowd, G., & Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, 16(2), 97–166.

Diaz, I. & Hollmén, J. (2002). Residual generation and visualization for understanding novel process conditions. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3 (pp. 2070–2075).: IEEE.

Diehl, C. P. & Hampshire, J. B. (2002). Real-time object classification and novelty detection for collaborative video surveillance. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3 (pp. 2620–2625).: IEEE.

Dunkels, A., Gronvall, B., & Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *In Proceedings of the First IEEE Workshop on Embedded Networked Sensors* Tampa, Florida, USA.

Edwards, W. K., Newman, M. W., Sedivy, J., Smith, T., & Izadi, S. (2002). Challenge: Recombinant Computing and the Speakeasy Approach. In *MobiCom'02* (pp. 279–286).

Endler, D. (1998). Applying machine learning to solaris audit data. In *Proceedings of the 1998 Annual Computer Security Application Conference* (pp. 268–279).

Ertöz, L., Steinbach, M., & Kumar, V. (2003). Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval*, 11, 83–103.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning*: Citeseer.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security* (pp. 77–101). Springer.

Eskinand, E. & Stolfo, S. (2001). Modeling system call for intrusion detection using dynamic window sizes. In *Proceedings of DARPA Information Survivability Conference and Exposition*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *et al.* (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (pp. 226–231).

Evans, D. (2011). The Internet of Things - How the Next Evolution of the Internet is Changing Everything. *CISCO white paper*, 1(April), 1–11.

Faisal, A. A., Selen, L. P., & Wolpert, D. M. (2008). Noise in the nervous system. *Nature reviews neuroscience*, 9(4), 292–303.

Fan, W., Miller, M., Stolfo, S., Lee, W., & Chan, P. (2004). Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, 6(5), 507–527.

Fawcett, T. & Provost, F. (1999). Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 53–62).: ACM.

Felleman, D. J. & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, N.Y. : 1991)*, 1(1), 1–47.

FlexSim Software Products, Inc. (1993). Flexsim simulation software. `https://www.flexsim.com/`. (accessed on 19 December 2016).

Floerkemeier, C., Roduner, C., & Lampe, M. (2007). Rfid application development with the accada middleware platform. *IEEE Systems Journal*, 1(2), 82–94.

Fortino, G., Guerrieri, A., Lacopo, M., Lucia, M., & Russo, W. (2013). An agent-based middleware for cooperating smart objects. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (pp. 387–398).: Springer.

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.

Fox, A. J. (1972). Outliers in time series. *Journal of the Royal Statistical Society. Series B (Methodological)*, (pp. 350–363).

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.

Galeano, P., Peña, D., & Tsay, R. S. (2006). Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, 101(474), 654–669.

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. & Hawkins, J. (2009). Towards a mathematical theory of cortical microcircuits. *PLoS Computational Biology*, 5(10).

Global Standards One (2003). Global standards one (gs1). `http://www.gs1.org`. (accessed on 15 February 2016).

GNU (1991). General public license, version 2. `https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html`. (accessed on 11 January 2017).

Golding, N. L., Jung, H.-y., Mickus, T., & Spruston, N. (1999). Dendritic calcium spike initiation and repolarization are controlled by distinct potassium channel subtypes in ca1 pyramidal neurons. *Journal of Neuroscience*, 19(20), 8789–8798.

Goldstein, M. (2014). *Anomaly Detection in Large Datasets*. Phd-thesis, University of Kaiserslautern, München, Germany.

Goldstein, M. & Dengel, A. (2012). Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track*, (pp. 59–63).

Goldstein, M. & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4), e0152173.

Goumopoulos, C. & Kameas, A. (2009). Smart Objects as Components of UbiComp Applications. *International Journal of Multimedia and Ubiquitous Engineering*, 4(3), 1–20.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, 11(1), 1–21.

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.

Guha, S., Rastogi, R., & Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5), 345–366.

Guinard, D. (2011). *A Web of Things Application Architecture - Integrating the Real-World into the Web*. PhD thesis, ETH Zurich.

Guttormsson, S. E., Marks, R., El-Sharkawi, M., & Kerszenbaum, I. (1999). Elliptical novelty grouping for on-line short-turn detection of excited running rotors. *IEEE Transactions on Energy Conversion*, 14(1), 16–22.

Gwadera, R., Atallah, M. J., & Szpankowski, W. (2005). Reliable detection of episodes in event sequences. *Knowledge and Information Systems*, 7(4), 415–437.

Hachem, S., Teixeira, T., & Issarny, V. (2011). Ontologies for the Internet of Things. *ACMIFIPUSENIX 12th International Middleware Conference*, (pp. 3:1–3:6).

Haller, S., Karnouskos, S., & Schroth, C. (2009). The internet of things in an enterprise context. In *Future Internet-FIS 2008* (pp. 14–28). Springer.

Han, D.-m. & Lim, J.-h. (2010). Smart home energy management system using IEEE 802.15.4 and zigbee.

Hardt, D. (2012). *The OAuth 2.0 Authorization Framework*. Technical report, RFC Editor.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Hautamaki, V., Karkkainen, I., & Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3 (pp. 430–433).: IEEE.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

Hawkins, J. (2014). The Science of Anomaly Detection. *Numenta, Inc.*

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

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

Hawkins, S., He, H., Williams, G., & Baxter, R. (2002). Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery* (pp. 170–180).: Springer.

Haykin, S. (2005). Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, 23.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9), 1641–1650.

He, Z., Xu, X., Huang, J. Z., & Deng, S. (2004). A frequent pattern discovery method for outlier detection. In *International Conference on Web-Age Information Management* (pp. 726–732).: Springer.

Healy, G. N., Clark, B. K., Winkler, E. A., Gardiner, P. A., Brown, W. J., & Matthews, C. E. (2011). Measurement of adults' sedentary time in population-based studies. *American journal of preventive medicine*, 41(2), 216–227.

Helal, S., Kim, E., & Hossain, S. (2010). Scalable approaches to activity recognition research. In *Proceedings of the 8th International Conference Pervasive Workshop* (pp. 450–453).

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.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., & Jansen, E. (2005). The gator tech smart house: A programmable pervasive space. *Computer*, 38(3), 50–60.

Heller, K. A., Svore, K. M., Keromytis, A. D., & Stolfo, S. J. (2003). One class support vector machines for detecting anomalous windows registry accesses. In *Proc. of the workshop on Data Mining for Computer Security*, volume 9.

Helman, P. & Bhangoo, J. (1997). A statistically based system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4), 449–466.

Hickinbotham, S. J. & Austin, J. (2000). Novelty detection in airframe strain data. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2 (pp. 536–539).: IEEE.

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

Hodge, V. J. & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2), 85–126.

Horn, P. S., Feng, L., Li, Y., & Pesce, A. J. (2001). Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry*, 47(12), 2137–2145.

Hu, W., Liao, Y., & Vemuri, V. R. (2003). Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning* (pp. 282–289).

Idé, T. & Kashima, H. (2004). Eigenspace-based anomaly detection in computer systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 440–449).: ACM.

Idé, T., Papadimitriou, S., & Vlachos, M. (2007). Computing correlation anomaly scores using stochastic nearest neighbors. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on* (pp. 523–528).: IEEE.

Intille, S. S., Larson, K., Tapia, E. M., Beaudin, J. S., Kaushik, P., Nawyn, J., & Rockinson, R. (2006). Using a live-in laboratory for ubiquitous computing research. In *International Conference on Pervasive Computing* (pp. 349–365).: Springer.

Intille, S. S., Rondoni, J., Kukla, C., Ancona, I., & Bao, L. (2003). A context-aware experience sampling tool. *CHI '03 extended abstracts on Human factors in computer systems - CHI '03*, (pp. 972).

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

Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Gerosa, M. A., & Hamida, A. B. (2011). Service-oriented middleware for the Future Internet: state of the art and research directions. *Journal of Internet Services and Applications*, 2(1), 23–45.

Jahromi, Z. F., Rajabzadeh, A., & Manashty, A. R. (2011). A Multi-Purpose Scenario-based Simulator for Smart House Environments. *arXiv preprint arXiv:1105.2902*, 9(1), 13–18.

Jain, G., Cook, D. J., & Jakkula, V. (2006). Monitoring Health by Detecting Drifts and Outliers for a Smart Environment. *International Conference On Smart homes and health Telematics*, (pp. 114–121).

Jakkula, V. & Cook, D. (2010). Outlier detection in smart environment structured power datasets. *Proceedings - 2010 6th International Conference on Intelligent Environments, IE 2010*, (pp. 29–33).

Jakkula, V. & Cook, D. J. (2008). Anomaly detection using temporal data mining in a smart home environment. *Methods of information in medicine*, 47(01), 70–75.

Jakkula, V. R. & Cook, D. J. (2011). Detecting anomalous sensor events in smart home data for enhancing the living experience. *Artificial intelligence and smarter living*, 11(201), 1.

Janakiram, D., Reddy, V., & Kumar, A. P. (2006). Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on* (pp. 1–6).: 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).

Jiang, L., Liu, D.-Y., & Yang, B. (2004). Smart home research. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 2 (pp. 659–663 vol.2).

Jiang, M.-F., Tseng, S.-S., & Su, C.-M. (2001). Two-phase clustering process for outliers detection. *Pattern recognition letters*, 22(6), 691–700.

Jin, W., Tung, A., Han, J., & Wang, W. (2006). Ranking outliers using symmetric neighborhood relationship. *Advances in Knowledge Discovery and Data Mining*, (pp. 577–593).

JME (2003). Java monkey engine. `http://www.jmonkeyengine.com`. (accessed on 26 November 2016).

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Jouve, W., Bruneau, J., & Consel, C. (2009). Diasim: A parameterized simulator for pervasive computing applications. In *2009 IEEE International Conference on Pervasive Computing and Communications* (pp. 1–3).

Kang, W., Shin, D., & Shin, D. (2010). Detecting and predicting of abnormal behavior using hierarchical Markov model in smart home network. *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference*, (pp. 410–414).

Karakovskiy, S. & Togelius, J. (2012). The Mario Ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4, 55–67.

Katz, B. F., Felinto, D. Q., Touraine, D., Poirier-Quinot, D., & Bourdot, P. (2015). Blendervr: Open-source framework for interactive and immersive vr. In *Virtual Reality (VR), 2015 IEEE* (pp. 203–204).: IEEE.

Kawsar, F. (2009). *A Document-Based Framework for User Centric Smart Object Systems*. PhD thesis, Waseda University.

Kawsar, F. & Nakajima, T. (2009). A Document Centric Framework for Building Distributed Smart Object Systems. In *2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing* (pp. 71–79). Tokyo: IEEE.

Kendall, K. (1999). *A database of computer attacks for the evaluation of intrusion detection systems*. Technical report, DTIC Document.

Kim, I., Park, H., Noh, B., Lee, Y., Lee, S., & Lee, H. (2006). Design and implementation of context-awareness simulation toolkit for context learning. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, volume 2 (pp. 96–103).: IEEE.

King, S., King, D., Astley, K., Tarassenko, L., Hayton, P., & Utete, S. (2002). The use of novelty detection techniques for monitoring high-integrity plant. In *Control Applications, 2002. Proceedings of the 2002 International Conference on*, volume 1 (pp. 221–226).: IEEE.

Knorr, E. M. & Ng, R. T. (1997). A unified approach for mining outliers. In *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research* (pp.1̃1).: IBM Press.

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB JournalThe International Journal on Very Large Data Bases*, 8(3-4), 237–253.

Kohonen, T., Schroeder, M., Huang, T., & Maps, S.-O. (2001). Springer-verlag new york. *Inc., Secaucus, NJ*, 43, 2.

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.

Kortuem, G., Kawsar, F., Fitton, D., & Sundramoorthy, V. (2010). Smart objects as building blocks for the Internet of things. *IEEE Internet Computing*, 14(1), 44–51.

Kou, Y., Lu, C.-T., & Chen, D. (2006). Spatial weighted outlier detection. In *Proceedings of the 2006 SIAM international conference on data mining* (pp. 614–618).: SIAM.

Kranz, M., Linner, T., Ellmann, B., Bittner, A., & Roalter, L. (2010a). Robotic service cores for ambient assisted living. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)* (pp. 1–8). IEEE.

Kranz, M., Roalter, L., & Michahelles, F. (2010b). Things that twitter: social networks and the internet of things. In *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)* (pp. 1–10).

Kranz, M., Schmidt, A., Rusu, R., Maldonado, A., Beetz, M., Hornler, B., & Rigoll, G. (2007). Sensing Technologies and the Player-Middleware for Context-Awareness in Kitchen Environments. In *Fourth International Conference on Networked Sensing Systems* (pp. 179–186).

Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009). Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 1649–1652).: ACM.

Kruegel, C. & Vigna, G. (2003). Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security* (pp. 251–261).: ACM.

Krügel, C., Toth, T., & Kirda, E. (2002). Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing* (pp. 201–208).: ACM.

Krzyska, C. (2006). *Smart House Simulation Tool*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark.

Kushalnagar, N., Montenegro, G., & Schumacher, C. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. *RFC4919*, 10, 1–13.

Lamme, V. A., Super, H., & Spekreijse, H. (1998). Feedforward, horizontal, and feedback processing in the visual cortex. *Current opinion in neurobiology*, 8(4), 529–535.

Lampesberger, H. (2016). Technologies for web and cloud service interaction: a survey. *Service Oriented Computing and Applications*, 10(2), 71–110.

Larkum, M. (2013). A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in neurosciences*, 36(3), 141–151.

Larkum, M. E. & Nevian, T. (2008). Synaptic clustering by dendritic signalling mechanisms. *Current opinion in neurobiology*, 18(3), 321–331.

Larkum, M. E., Zhu, J. J., & Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725), 338–341.

Lauer, M. (2001). A mixture approach to novelty detection using training data with outliers. In *European Conference on Machine Learning* (pp. 300–311).: Springer.

Laurikkala, J., Juhola, M., Kentala, E., Lavrac, N., Miksch, S., & Kavsek, B. (2000). Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, volume 1 (pp. 20–24).

Lavin, A. & Ahmad, S. (2015). Evaluating Real-time Anomaly Detection Algorithms - the Numenta Anomaly Benchmark. *14th International Conference on Machine Learning and Applications (IEEE ICMLA'15)*.

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining* (pp. 25–36).: SIAM.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

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, J. W., Helal, A., Sung, Y., & Cho, K. (2013). A context-driven approach to scalable human activity simulation. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation* (pp. 373–378).: ACM.

Lei, Z., Yue, S., Yu, C., & Yuanchun, S. (2010). Shsim: An osgi-based smart home simulator. In *3rd IEEE International Conference on Ubi-media Computing (U-Media)* (pp. 87–90).: IEEE.

Leiba, B. (2012). OAuth Web Authorization Protocol.

Lertlakkhanakul, J., Choi, J. W., & Kim, M. Y. (2008). Building data model and simulation platform for spatial interaction management in smart home. *Automation in Construction*, 17(8), 948–957.

Lewis, D. D. (1997). Reuters-21578 text categorization test collection, distribution 1.0. *http://www. research. att. com/˜ lewis/reuters21578. html*.

Li, H., Chen, Y., & He, Z. (2012). The Survey of RFID Attacks and Defenses. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)* (pp. 1–4).: IEEE.

Lim, B. & Dey, A. (2010). Toolkit to support intelligibility in context-aware applications. In *the 12th ACM international conference on Ubiquitous computing - Ubicomp '10* (pp. 13–22). New York, New York, USA: ACM Press.

Lin, C.-H., Ho, P.-H., & Lin, H.-C. (2014). Framework for NFC-Based Intelligent Agents: A Context-Awareness Enabler for Social Internet of Things. *International Journal of Distributed Sensor Networks*, 2014, 1–16.

Lin, J., Keogh, E., Fu, A., & Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. In *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on* (pp. 329–334).: IEEE.

Lindner, W. L. W. & Meier, J. M. J. (2006). Securing the Borealis Data Stream Engine. *2006 10th International Database Engineering and Applications Symposium (IDEAS'06)*.

Liu, H. L. H., Bolic, M., Nayak, A., & Stojmenovic, I. (2008). Taxonomy and Challenges of the Integration of RFID and Wireless Sensor Networks. *IEEE Network*, 22.

LLC, S. (2006). Simio simulation software. `http://www.simio.com/`. (accessed on 19 December 2016).

Lu Tan & Wang, N. (2010). Future internet: The Internet of Things. *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 5, V5–376–V5–380.

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.

Ma, J. & Perkins, S. (2003a). Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 613–618).: ACM.

Ma, J. & Perkins, S. (2003b). Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3 (pp. 1741–1745).: IEEE.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9), 1659–1671.

Mahoney, M. V. & Chan, P. K. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 376–385).: ACM.

Mahoney, M. V. & Chan, P. K. (2003). Learning rules for anomaly detection of hostile network traffic. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on* (pp. 601–604).: IEEE.

Mainetti, L., Patrono, L., & Vilei, A. (2011). Evolution of Wireless Sensor Networks towards the Internet of Things : a Survey. *Software, Telecommunications and Computer Networks (SoftCOM)*, (pp. 2–7).

Major, G., Larkum, M. E., & Schiller, J. (2013). Active properties of neocortical pyramidal neuron dendrites. *Annual review of neuroscience*, 36, 1–24.

Major, G., Polsky, A., Denk, W., Schiller, J., & Tank, D. W. (2008). Spatiotemporally graded nmda spike/plateau potentials in basal dendrites of neocortical pyramidal neurons. *Journal of neurophysiology*, 99(5), 2584–2601.

Manevitz, L. M. & Yousef, M. (2001). One-class svms for document classification. *Journal of Machine Learning Research*, 2(Dec), 139–154.

Manson, G. (2002). Identifying damage sensitive, environment insensitive features for damage detection. In *Proceedings of the third international conference on identification in engineering systems* (pp. 187–197).

Manson, G., Pierce, G., & Worden, K. (2001). On the long-term stability of normal condition for damage detection in a composite panel. In *Key Engineering Materials*, volume 204 (pp. 359–370).: Trans Tech Publ.

Manson, G., Pierce, S. G., Worden, K., Monnier, T., Guy, P., & Atherton, K. (2000). Long-term stability of normal condition data for novelty detection. In *SPIE's 7th Annual International Symposium on Smart Structures and Materials* (pp. 323–334).: International Society for Optics and Photonics.

MANTENUTO, G. (2013). Simulation of ambient sensors in a robotic home for elderly people.

Marceau, C. (2001). Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the 2000 workshop on New security paradigms* (pp. 101–110).: ACM.

Martin, K. A. & Schröder, S. (2013). Functional heterogeneity in neighboring neurons of cat primary visual cortex in response to both artificial and natural stimuli. *Journal of Neuroscience*, 33(17), 7325–7344.

May, T. (2011). *Social research*. McGraw-Hill Education (UK).

McCarthy, J. (1998). What is Artificial Intelligence? a note.

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.

McGlinn, K., O'Neill, E., Gibney, A., O'Sullivan, D., & Lewis, D. (2010). Simcon: A tool to support rapid evaluation of smart building application design using context simulation and virtual reality. *J. UCS*, 16(15), 1992–2018.

Medaglia, C. M. & Serbanati, A. (2010). An overview of privacy and security issues in the internet of things. In D. Giusto, A. Iera, G. Morabito, & L. Atzori (Eds.), *The Internet of Things* (pp. 389–395). New York, NY: Springer.

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

Messer, a., Kunjithapatham, A., Sheshagiri, M., & Kumar, P. (2006). InterPlay: A Middleware for Seamless Device Integration and Task Orchestration in a Networked Home. *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*, 2(1), 296–307.

Miche, M., Schreiber, D., & Hartmann, M. (2009). Core Services for Smart Products. *Architecture*, (pp. 1–4).

Miller, Z., Dickinson, B., Deitrick, W., Hu, W., & Wang, A. H. (2014). Twitter spammer detection using data stream clustering. *Information Sciences*, 260, 64–73.

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.

Mohamed, N. & Al-Jaroodi, J. (2011). A survey on service-oriented middleware for wireless sensor networks. *Service Oriented Computing and Applications*, 5(2), 71–85.

Molisch, A. F., Balakrishnan, K., Chong, C.-C., Emami, S., Fort, A., Karedal, J., Kunisch, J., Schantz, H., Schuster, U., & Siwiak, K. (2004). Ieee 802.15. 4a channel model-final report. *IEEE P802*, 15(04), 0662.

Molla, M. & Ahamed, S. (2006). A survey of middleware for sensor network and challenges. In *2006 International Conference on Parallel Processing Workshops (ICPPW'06)* (pp. 223–228).: IEEE.

Munguia Tapia, E. (2003). *Activity recognition in the home setting using simple and ubiquitous sensors*. PhD thesis, Massachusetts Institute of Technology.

Murray, A. F. (2001). Novelty detection using products of simple expertsa potential architecture for embedded systems. *Neural Networks*, 14(9), 1257–1264.

Mylonas, A., Kastania, A., & Gritzalis, D. (2013). Delegate the smartphone user ? Security awareness in smartphone platforms. *Computers and Security*.

Mylonas, A., Theoharidou, M., & Gritzalis, D. (2014). Assessing privacy risks in android: A user-centric approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8418 LNCS, 21–37.

Newman, I. & Benz, C. R. (1998). *Qualitative-quantitative research methodology: Exploring the interactive continuum*. SIU Press.

Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., & Ito, M. (2006). Ubireal: realistic smartspace simulator for systematic testing. In *International Conference on Ubiquitous Computing* (pp. 459–476).: Springer.

Noh, S.-K., Kim, Y.-M., Kim, D., & Noh, B.-N. (2006). Network Anomaly Detection Based on Clustering of Sequence Patterns. *Computational Science and Its Applications - ICCSA 2006*, (pp. 349–358).

Noury, N., Poujaud, J., Fleury, A., Nocua, R., Haddidi, T., & Rumeau, P. (2011). Smart sweet home a pervasive environment for sensing our daily activity? In *Activity recognition in pervasive intelligent environments* (pp. 187–208). Springer.

Novak, M., Binas, M., & Jakab, F. (2012). Unobtrusive anomaly detection in presence of elderly in a smart-home environment. *Proceedings of 9th International Conference, ELEKTRO 2012*, (pp. 341–344).

Nugent, C., Mulvenna, M., Hong, X., & Devlin, S. (2009). Experiences in the development of a smart lab. *International Journal of Biomedical Engineering and Technology*, 2(4), 319–331.

O'Brien, J., Rodden, T., & Hughes, J. (1999). At home with the technology: an ethnographic study of a set-top-box trial. *ACM Transactions on Computer Human Interaction*, 6(3), 282–308.

Olsen Jr, D. R., Nielsen, S. T., & Parslow, D. (2001). Join and capture: a model for nomadic interaction. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (pp. 131–140).: ACM.

Olshausen, B. A. & Field, D. J. (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4), 481–487.

O'Neill, E., Klepal, M., Lewis, D., O'Donnell, T., O'Sullivan, D., & Pesch, D. (2005). A testbed for evaluating human interaction with ubiquitous computing environments. In *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities* (pp. 60–69).: IEEE.

OpenGL (1992). Opengl. `https://www.opengl.org`. (accessed on 05 November 2016).

Orr, R. J. & Abowd, G. D. (2000). The Smart Floor : A Mechanism for Natural User Identification and Tracking. *Conference on Human Factors in Computing Systems,*, (pp. 275–276).

Otey, M., Parthasarathy, S., Ghoting, A., Li, G., Narravula, S., & Panda, D. (2003). Towards nic-based intrusion detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 723–728).: ACM.

Otey, M. E., Ghoting, A., & Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data mining and knowledge discovery*, 12(2-3), 203–228.

Palshikar, G. K. (2005). Distance-based outliers in sequences. In *International Conference on Distributed Computing and Internet Technology* (pp. 547–552).: Springer.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., & Faloutsos, C. (2003). Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on* (pp. 315–326).: IEEE.

Park, B., Min, H., Bang, G., & Ko, I. (2015). The User Activity Reasoning Model in a Virtual Living Space Simulator. *International Journal of Software Engineering and Its Applications*, 9(6), 53–62.

Park, J., Moon, M., Hwang, S., & Yeom, K. (2007). CASS: A context-aware simulation system for smart home. In *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA)* (pp. 461–467).: IEEE.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3), 1065–1076.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1), 414–454.

Petreanu, L., Mao, T., Sternson, S. M., & Svoboda, K. (2009). The subcellular organization of neocortical excitatory connections. *Nature*, 457(7233), 1142–1145.

Phua, C., Alahakoon, D., & Lee, V. (2004). Minority report in fraud detection: classification of skewed data. *Acm sigkdd explorations newsletter*, 6(1), 50–59.

Pires, A. & Santos-Pereira, C. (2005). Using clustering and robust estimators to detect outliers in multivariate data. In *Proceedings of the International Conference on Robust Statistics*.

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

Poirazi, P., Brannon, T., & Mel, B. W. (2003). Pyramidal neuron as two-layer neural network. *Neuron*, 37(6), 989–999.

Pokrajac, D., Lazarevic, A., & Latecki, L. J. (2007). Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on* (pp. 504–515).: IEEE.

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.

Polsky, A., Mel, B. W., & Schiller, J. (2004). Computational subunits in thin dendrites of pyramidal cells. *Nature neuroscience*, 7(6), 621–627.

Portnoy, L., Eskin, E., & Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*: Citeseer.

Preuveneers, D. & Berbers, Y. (2008). Internet of things: A context-awareness perspective. In *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems* (pp. 287–308). Auerbach Publications.

Purdy, S. (2016). Encoding data for HTM systems. *CoRR*, abs/1602.05925.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 32 (pp. 151–170).

Rabiner, L. & Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1), 4–16.

Rah, J.-C., Bas, E., Colonell, J., Mishchenko, Y., Karsh, B., Fetter, R. D., Myers, E. W., Chklovskii, D. B., Svoboda, K., Harris, T. D., *et al.* (2013). Thalamocortical input onto layer 5 pyramidal neurons measured using quantitative large-scale array tomography. *Frontiers in neural circuits*, 7, 177.

Ramadas, M., Ostermann, S., & Tjaden, B. (2003). Detecting anomalous network traffic with self-organizing maps. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 36–54).: Springer.

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29 (pp. 427–438).: ACM.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., & Clarke, S. (2016). Middleware for internet of things: A survey. *IEEE Internet of Things Journal*, 3(1), 70–95.

Rivera, J. & Meulen, R. (2014). Sales of Smartphones Grew 20 Percent in Third Quarter of 2014.

Roalter, L., Kranz, M., & Möller, A. (2010). A middleware for intelligent environments and the internet of things. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6406 LNCS (pp. 267–281).

Roberts, S. & Tarassenko, L. (1994). A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6(2), 270–284.

Roberts, S. J. (2000). Extreme value statistics for novelty detection in biomedical data processing. *IEE Proceedings-Science, Measurement and Technology*, 147(6), 363–367.

Rodden, T. & Benford, S. (2003). The evolution of buildings and implications for the design of ubiquitous domestic environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 9–16).: ACM.

Rodner, T. & Litz, L. (2013). Data-driven generation of rule-based behavior models for an ambient assisted living system. In *IEEE Third International Conference on Consumer Electronics* (pp. 35–38).: IEEE.

Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., & Nahrstedt, K. (2002). A middleware infrastructure for active spaces.

Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10), 2266–2279.

Roth, V. (2006). Kernel fisher discriminants for outlier detection. *Neural computation*, 18(4), 942–960.

Rousseeuw, P. J. & Leroy, A. M. (2005). *Robust regression and outlier detection*, volume 589. John wiley & sons.

Roy, N., Roy, A., & Das, S. K. (2006). Context-Aware Resource Management in Multi-Inhabitant Smart Homes : A Nash H -Learning based Approach. *Proceedings - Fourth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2006*, 2006, 148–158.

Russell, S. & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach, 3rd edition*. Prentice Hall.

Sajda, P., Spence, C., & Parra, L. (2003). A multi-scale probabilistic network model for detection, synthesis and compression in mammographic image analysis. *Medical image analysis*, 7(2), 187–204.

Salvador, S., Chan, P., & Brodie, J. (2004). Learning states and rules for time series anomaly detection. In *FLAIRS Conference* (pp. 306–311).

Sarawagi, S., Agrawal, R., & Megiddo, N. (1998). Discovery-driven exploration of olap data cubes. In *International Conference on Extending Database Technology* (pp. 168–182).: Springer.

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

Saunders, M. N. (2011). *Research methods for business students, 5/e*. Pearson Education India.

Saxena, N., Roy, A., & Shin, J. (2007). Chase: Context-aware heterogenous adaptive smart environments using optimal tracking for residents comfort. In *International Conference on Ubiquitous Intelligence and Computing* (pp. 133–142).: Springer.

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

Schiller, J., Major, G., Koester, H. J., & Schiller, Y. (2000). Nmda spikes in basal dendrites of cortical pyramidal neurons. *Nature*, 404(6775), 285–289.

Schiller, J. & Schiller, Y. (2001). Nmda receptor-mediated dendritic spikes and coincident signal amplification. *Current opinion in neurobiology*, 11(3), 343–348.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7), 1443–1471.

Searle, J. R. (1980). Minds, Brains, and Programs. *Behavioral and Brain Sciences*, 3, 1–19.

Sebyala, A. A., Olukemi, T., Sacks, L., & Sacks, D. L. (2002). Active platform security through intrusion detection using naive bayesian network for anomaly detection. In *London Communications Symposium*: Citeseer.

Shekhar, S., Lu, C.-T., & Zhang, P. (2002). Detecting graph-based spatial outliers. *Intelligent Data Analysis*, 6(5), 451–468.

Shewhart, W. A. (1931). *Economic control of quality of manufactured product*. ASQ Quality Press.

Shin, J. H., Lee, B., & Park, K. S. (2011). Detection of abnormal living patterns for elderly living alone using support vector data description. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 15(3), 438–448.

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., & Chang, L. (2003). *A novel anomaly detection scheme based on principal component classifier*. Technical report, DTIC Document.

Singh, S. & Markou, M. (2004). An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, 16(4), 396–407.

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

Smith, R., Bivens, A., Embrechts, M., Palagiri, C., & Szymanski, B. (2002). Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, (pp. 579–584).

Solberg, H. E. & Lahti, A. (2005). Detection of outliers in reference distributions: performance of horns algorithm. *Clinical chemistry*, 51(12), 2326–2332.

Soma Bandyopadhyay, Sengupta, M., Maiti, S., & Dutta, S. (2011). Role Of Middleware For Internet Of Things: A Study. *International Journal of Computer Science & Engineering Survey*, 2(3), 94–105.

Song, Q., Hu, W., & Xie, W. (2002). Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4), 440–448.

Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3), 206–221.

Srivastava, A. N. & Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace conference, 2005 IEEE* (pp. 3853–3862).: IEEE.

Srivastava, L. (2006). Pervasive, ambient, ubiquitous: the magic of radio. In *European Commission Conference From RFID to the Internet of Things, Bruxelles, Belgium*.

Stahl, C. & Schwartz, T. (2010). Modeling and simulating assistive environments in 3-d with the yamamoto toolkit. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (pp. 1–6).: IEEE.

Stefansky, W. (1972). Rejecting outliers in factorial designs. *Technometrics*, 14(2), 469–479.

Stuart, G. J. & Häusser, M. (2001). Dendritic coincidence detection of epsps and action potentials. *Nature neuroscience*, 4(1), 63–71.

Sun, J., Xie, Y., Zhang, H., & Faloutsos, C. (2007). Less is more: Compact matrix representation of large sparse graphs. In *Proceedings of 7th SIAM International Conference on Data Mining*.

Sun, P. & Chawla, S. (2004). On local spatial outliers. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (pp. 209–216).: IEEE.

Sung, Y., Helal, A., Lee, J. W., & Cho, K. (2013). Bayesian-based Scenario Generation Method for Human Activities. *ACM SIGSIM conference on Principles of advanced discrete simulation (SIGSIM-PADS)*, (pp. 147–157).

Surace, C., Worden, K., *et al.* (1998). A novelty detection method to diagnose damage in structures: an application to an offshore platform. In *The Eighth International Offshore and Polar Engineering Conference*: International Society of Offshore and Polar Engineers.

Suzuki, E., Watanabe, T., Yokoi, H., & Takabayashi, K. (2003). Detecting interesting exceptions from medical test data with visual summarization. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on* (pp. 315–322).: IEEE.

Synnott, J., Chen, L., Nugent, C., & Moore, G. (2014). The creation of simulated activity datasets using a graphical intelligent environment simulation tool. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE* (pp. 4143–4146).: IEEE.

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

Synnott, J., Nugent, C., Zhang, S., Calzada, A., Cleland, I., Espinilla, M., Quero, J. M., & Lundstrom, J. (2016). Environment simulation for the promotion of the open data initiative. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)* (pp. 1–6).

Tan, L. & Wang, N. (2010). Future internet: The Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5 (pp. 376–380). Chengdu.

Tan, P.-N. *et al.* (2006). *Introduction to data mining*. Pearson Education India.

Tandon, G. & Chan, P. K. (2007). Weighting versus pruning in rule validation for detecting network and host anomalies. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 697–706).: ACM.

Tang, J., Chen, Z., Fu, A. W.-C., & Cheung, D. W. (2002). Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 535–548).: Springer.

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.

Tarassenko, L., Hayton, P., Cerneaz, N., & Brady, M. (1995). Novelty detection for the identification of masses in mammograms. In *1995 Fourth International Conference on Artificial Neural Networks* (pp. 442–447).: IET.

Taylor, O. & Addison, D. (2000). Novelty detection using neural network technology. In *COMADEM 2000: 13 th International Congress on Condition Monitoring and Diagnostic Engineering Management* (pp. 731–743).

Thoma, M. (2014). `https://martin-thoma.com/twiddle/`.

Thompson, B. B., Marks, R. J., Choi, J. J., El-Sharkawi, M. A., Huang, M.-Y., & Bunje, C. (2002). Implicit learning in autoencoder novelty assessment. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3 (pp. 2878–2883).: IEEE.

Trifa, M. V. (2011). *Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks*. PhD thesis, ETH Zurich.

Tsay, R. S., Peña, D., & Pankratz, A. E. (2000). Outliers in multivariate time series. *Biometrika*, 87(4), 789–804.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, 59(236), 433–460.

Unity3D (2005). Unity3d. `https://unity3d.com/`. (accessed on 22 November 2016).

Uusitalo, M. A. (2006). Global vision for the future wireless world from the wwrf. *IEEE Vehicular Technology Magazine*, 1(2), 4–8.

Valdes, A. & Skinner, K. (2000). Adaptive, model-based monitoring for cyber attack detection. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 80–93).: Springer.

van Berlo, A., Bob, A., Jan, E., Klaus, F., Maik, H., & Charles, W. (1999). Design guidelines on smart homes. *A COST 219bis Guidebook (October 1999)*.

Van Nguyen, T., Kim, J. G., & Choi, D. (2009). Iss: the interactive smart home simulator. In *11th International Conference on Advanced Communication Technology (ICACT)*, volume 3 (pp. 1828–1833).: IEEE.

Van Phuong, T., Hung, L. X., Cho, S. J., Lee, Y.-K., & Lee, S. (2006). An anomaly detection algorithm for detecting attacks in wireless sensor networks. In *International Conference on Intelligence and Security Informatics* (pp. 735–736).: Springer.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vermesan, O., Peter, F., Patrick, G., Sergio, G., Harald, Sundmaeker Alessandro, B., Ignacio Soler, J., Margaretha, M., Mark, H., Markus, E., & Pat, D. (2011). Internet of Things: Strategic Research Roadmap. In *Internet of Things-Global Technological and Societal Trends* (pp. 9–52). River Publishers.

Vinje, W. E. & Gallant, J. L. (2002). Natural stimulation of the nonclassical receptive field increases information transmission efficiency in v1. *Journal of Neuroscience*, 22(7), 2904–2915.

Von Neumann, J. (1993). First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4), 27–75.

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural computation*, 1(1), 39–46.

WebGL (2006). Webgl. `https://www.khronos.org/webgl/`. (accessed on 05 November 2016).

Wei, L., Qian, W., Zhou, A., Jin, W., & Jeffrey, X. Y. (2003). Hot: Hypergraph-based outlier test for categorical data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 399–410).: Springer.

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), 48–55.

Williams, G., Baxter, R., He, H., Hawkins, S., & Gu, L. (2002). A comparative study of rnn for outlier detection in data mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* (pp. 709–712).: IEEE.

Williams, P. L. *et al.* (1980). *Gray's anatomy*. London: Churchill Livingstone, 1980.

Wong, W.-K., Moore, A., Cooper, G., & Wagner, M. (2003). Bayesian network anomaly pattern detection for disease outbreaks. In *ICML* (pp. 808–815).

Xie, M., Hu, J., & Tian, B. (2012). Histogram-based online anomaly detection in hierarchical wireless sensor networks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on* (pp. 751–759).: IEEE.

Xu, W. X. W., Xin, Y. X. Y., & Lu, G. L. G. (2007). A System Architecture for Pervasive Computing. *Third International Conference on Natural Computation (ICNC 2007)*, 5.

Yairi, T., Kato, Y., & Hori, K. (2001). Fault detection by mining association rules from house-keeping data. In *Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 3: Citeseer.

Yamanishi, K. & Takeuchi, J.-i. (2001). Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 389–394).: ACM.

Yamanishi, K., Takeuchi, J.-I., Williams, G., & Milne, P. (2000). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 320–324).: ACM.

Yamazaki, T. (2007). The ubiquitous home. *International Journal of Smart Home*, 1(1), 17–22.

Yan, L., Zhang, Y., Yang, L. T., & Ning, H. (2008). *The Internet of things: from RFID to the next-generation pervasive networked systems*. Auerbach Publications.

Ye, N. & Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2), 105–112.

Yen, S.-C., Baker, J., & Gray, C. M. (2007). Heterogeneity in the responses of adjacent neurons to natural stimuli in cat striate cortex. *Journal of neurophysiology*, 97(2), 1326–1341.

Yeung, D.-Y. & Chow, C. (2002). Parzen-window network intrusion detectors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4 (pp. 385–388).: IEEE.

Yoshimura, Y., Sato, H., Imamura, K., & Watanabe, Y. (2000). Properties of horizontal and vertical inputs to pyramidal cells in the superficial layers of the cat visual cortex. *Journal of Neuroscience*, 20(5), 1931–1940.

Youngblood, G. M., Cook, D. J., & Holder, L. B. (2005). Seamlessly engineering a smart environment. In *SMC* (pp. 548–553).

Zhang, D., Yang, L. T., & Huang, H. (2011). Searching in Internet of Things: Vision and Challenges. *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications*, (pp. 201–206).

Zhang, J. & Wang, H. (2006a). Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, 10(3), 333–355.

Zhang, L. Z. L. & Wang, Z. W. Z. (2006b). Integration of RFID into Wireless Sensor Networks: Architectures, Opportunities and Challenging Problems. *2006 Fifth International Conference on Grid and Cooperative Computing Workshops*.

ZigBee, A. (2006). Zigbee specification. *ZigBee document 053474r13*.