Application of the Hierarchical Temporal Memory to Smart Transport

By

Afaf Almehmadi

A Thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy at

Staffordshire University, Stafford, United Kingdom

November 2022

Abstract

Traffic congestion has a significant effect on the economy and environment. There is a need for more innovative solutions that adopt new technologies to tackle the congestion issue to reduce journey times and fuel emissions. Recently, there have been many attempts to address this issue using different machine learning approaches such as the k-nearest neighbour algorithm and artificial neural network. However, additional measures and new methods need to be taken and investigated respectively to resolve the traffic congestion issue.

Due to new advanced IoT technologies, the data generated by sensors are huge and continually changing. Thus, there is a need for a machine learning algorithm that is capable of dealing with this huge amount of data. One of the new advancements in machine learning models is the Hierarchical Temporal Memory (HTM), a neuroscience-based algorithm that attempts to mimic the neocortex functions in the human brain. This made HTM a good choice since the memory requirements for the HTM algorithm are less than compared to machine learning algorithms.

To the author's knowledge, using HTM for the smart transport application is not commonly adopted in the literature, and more research is needed. Therefore, a proposal of an HTM theory-based framework is presented in this work. The novelty of this framework is the use of multi-level anomaly detection with a predictive feedback network. Each level comprises different inputs that are combined to an upper level to detect abnormalities from previous patterns in the lower levels. It then introduces a predictive level model to provide feedback by using the patterns learnt from the combined upper levels of the HTM hierarchy.

To evaluate and test the proposed framework, several synthetic and real data sets were reviewed. Based on the results of the review, a pre-processing part of the Highways England's Motorway Incident Detection and Automatic Signalling (MIDAS) traffic database was required to evaluate the performance of the proposed framework.

The framework was implemented using the latest Numenta Platform for Intelligent Computing (NuPIC), which is an implementation of HTM. The framework was evaluated using two sets of input data from the pre-processed MIDAS datasets. The parameters of the encoders, spatial pooler and temporal memory were optimised to produce the highest accuracy and F-measure. The results show a rise in both accuracy and F-measure by 2% and 6.5% respectively.

The proposed framework performance is evaluated against a number of conventional anomaly detection methods using F-measure as one of the performance metrics. The analysis of the results revealed that the proposed framework achieved a 60% F-measure, which outperforms the K-Nearest Neighbour Global Anomaly Score (KNN-GAS) by 35%, the Independent Component Analysis-Local Outlier Probability (ICA-LoOP) by 34% and the Singular Value Decomposition Influence Outlier (SVD-IO) by 33%. These results provide evidence that the proposed HTM based framework can achieve better performance when dealing with traffic detection datasets.

Acknowledgements

I would like to thank my supervisor Dr Tomasz Bosakowski for his dedicated supervision during my PhD study. His endless help, kindness, patience, and thoughtful considerations are greatly appreciated.

I would like to thank my family for their unlimited support which made my PhD journey possible, especially, my amazing mother and father who blessed me every step of my life. My husband and beloved son and daughter who accompanied me throughout this stage of my life.

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

I thank all those who stood beside me during my PhD study during Covid-19 pandemic.

Publications

Almehmadi, Afaf, et al. "Htm based anomaly detecting model for traffic congestion." *Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing*. 2020.

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
ANPR	**A**utomatic **N**umber **P**late **R**ecognition
ATMs	**A**utomated **T**eller **M**achines
BN	**B**ayes **N**etworks
CCTV	**C**losed **C**ircuit **T**elevision
CLA	**C**ortical **L**earning **A**lgorithm
CPU	**C**entral **P**rocessing **U**nit
DRIVENET	**D**igital **R**oadway **I**nteractive **V**isualization and **E**valuation **Net**work
GA	**G**enetic **A**lgorithm
GIS	**G**eographic **I**nformation **S**ystem
GPS	**G**eographic **P**oint **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
HATMS	**H**ighways **E**ngland **T**raffic **M**anagement **S**ystems
HTM	**H**ierarchical **T**emporal **M**emory
ICA-LoOP	**I**ndependent **C**omponent **A**nalysis**-Lo**cal **O**utlier **P**robability
IST	**I**ntelligent **T**ransport **S**ystems
IoT	**I**nternet **o**f **T**hings
KNN	**K-N**earest **N**eighbours
KNN-GAS	**K-N**earest **N**eighbor **G**lobal **A**nomaly **S**core
ML	**M**achine **L**earning
MIDAS	**M**otorway **I**ncident **D**etection and **A**utomatic **S**ignalling
NTIS	**N**ational **T**raffic **I**nformation **S**ervice
NuPIC	**N**umenta **P**latform for **I**ntelligent **C**omputing
NN	**N**euron **N**etworks
PCA	**P**rincipal **C**omponent **A**nalysis
PSO	**P**article **S**warm **O**ptimisation
RFID	**R**adio **F**requency **Id**entification
SCATS	**S**ydney **C**oordinated **A**daptive **T**raffic **S**ystem
SDR	**S**parse **D**istributed **R**epresentations

SVD-IO	**S**ingular **V**alue **D**ecomposition **I**nfluence **O**utlier
SVM	**S**upport **V**ector **M**achine
STS	**S**mart **T**ransport **S**ystems
SP	**S**patial **P**ooler
SUMO	**S**imulation **o**f **U**rban **M**obility
TM	**T**emporal **M**emory
VMS	**V**ariable **M**essage **S**ign

List of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

According to Fleming (2019), the total cost of traffic congestion in the US in 2018 was around $87 billion. With the expectation of the world's population to be rise by 68% by 2050, the existing transportation systems must attain goals that ensure less congestion and minimal environmental effect (Fleming, 2019). Furthermore, the expected increment in the cost of traffic congestion in the UK by 2030 will exceed £21 Billion with a more than 63% growth compared to the cost reported in 2013 by to INRIX and the Centre for Economics and Business Research[1].

The inclusive aggregate effect of the congestion is projected to be more than £300 billion by 2030, including £191 billion in costs to the economy due to fuel consumption from spending more time on roads rather than work. The report also highlighted a total cost of £115 billion that affects business related to the vehicle and transport sectors, and these effects would increase individuals' household bills (Simmons, 2014). The study also predicts an increment in the need for road usage around the UK in the upcoming 16 years. This increase will contribute to the rise of congestion and will be very costly to the economy (Simmons, 2014). Therefore, there is a need for more innovative solutions that adopt new technologies, such as the Internet of Things (IoT), to produce smart transport to overcome old transportation planning approaches and their related issues.

[1] https://inrix.com/press-releases/traffic-congestion-to-cost-the-uk-economy-more-than-300-billion-over-the-next-16-years/

The IoT offers technologies for optimising private and public transportation (Syed et al., 2022). Vehicles can be equipped with Wi-Fi or other sensors to connect to the Internet. Zantalis et al. (2019) pointed out that the new emergence of the IoT technology has introduced many advantages that provide innovative ways to make transportation more intelligent and increase the abilities of its infrastructure to cope with these changes. The main goals of the application of IoT to transportation are to make roads safer, optimise routes, reduce infrastructure costs, and reduce traffic congestion (Cook et al., 2020; Nižetić et al., 2020).

In the transport sector, efforts by administrators to produce more reliable economic and social benefits are not guaranteed, as they face many difficulties in attempting to decrease traffic congestion (Karuppanagounder & Muneera, 2018). Detecting abnormalities as early as possible is vital for transport sector administrators. Early detection ensures seamless flow, reduces travel times, and reduces fuel emissions. Detecting road anomalies in smart transportation becomes vital as road conditions directly impact various aspects of the transportation domain (Zantalis et al., 2019). Although the aid of the IoT sensors makes smart transport more context-aware systems, it gives rise to more complicated issues in the systems related to the data collected (Khan, 2017). For example, data from traffic monitoring sensors change rapidly because of the movement of vehicles. Thus, such data must be processed quickly to provide reliable traffic applications to its users (Khan, 2017).

Recently, there have been many attempts to address the issue of the detection of road anomalies using machine learning approaches. Machine learning is defined as

"*a sort of artificial intelligence (AI) that enables computers to learn without being* *explicitly programmed*" (Kashyap, 2017).

Furthermore, machine learning is a method of data analysis that uses an algorithm to learn from data iteratively. Machine learning techniques enable computers to find unseen insights without having been explicitly programmed where to search. In addition, applications that use machine learning techniques can process new data once the software has been trained (Alpaydın, 2014).

However, there is a doubt that the current state-of-the-art machine learning techniques detect anomalies in the data stream (Dogra & Kaur, 2022). Thus, there is a need to investigate new methods and algorithms, such as bio-inspired algorithms that employ a higher level of intelligence and are inspired by biological mechanisms (Del Ser et al., 2020). There is a limited number of publications related to bio-inspired algorithms applied to smart transport. A few authors have considered the benefits of biologically inspired Artificial Neural Networks (ANNs) to examine traffic conditions. For instance, a study by (Goves et al., 2016a) uses data provided by Highways England's Motorway Incident Detection and Automatic Signalling (MIDAS) system to forecast traffic conditions in 15 minutes. The study showed that using short-term data from multiple sources was more effective in traffic forecasting than using data collected from one source at a specific time. However, the main disadvantage of the proposed model is that it needs to be retrained each time that the model is applied to other roads, which is a time-consuming process. Passow et al. (2013) use ANN as a forecaster methodology that includes the stops during a journey and waiting time to predict traffic flow conditions (Passow et al., 2013).

In another study, a bio-inspired ant-based algorithm was utilised to mimic the food search behaviour of ants. The proposed model is ‘a multi-objective ant-based vehicle congestion avoidance system’ that can reduce congestion in crowded areas. The main objectives of this study are to find the shortest path for the vehicle to use with minimal congestion (Jabbarpour et al., 2014).

These studies provide important insights into traffic congestion management solutions, yet they still have weaknesses. For example, a common fault of the ANN-based model, is it must be retrained every time the model runs. Moreover, traffic congestion management is a real-time application, which means that the system depends on processing data streams from various sensors (Lutkevich, 2022). These sensors are integrated IoT technologies such as speed sensors, loop detectors, etc. In this way, the streams of speed, density and data are used to detect unexpected events that could affect road conditions quickly and help to resolve issues before the event is reported (Figueiras et al., 2018). These functions require an online system designed to extract and store relevant information from various types of data, regardless of their origins, i.e., the system is not designed to be trained once or to work on only one class of data (Zyarah, 2015).

An example of the new advancements in Machine Learning models is Hierarchical Temporal Memory (HTM). An HTM algorithm is a neuroscience-based algorithm that mimics the capabilities of the neocortex in the human brain (Hawkins & Ahmad, 2016). The authors claim that its computational models can achieve or exceed the production level of the brain for intellectual tasks (Hawkins & George, 2006). Furthermore, Numenta, which is a part of developing machine intelligence through neocortical theory states that applications that include data streaming over time can suit HTM algorithms. In addition, any application that has an inherent data structure would suit HTM algorithms. Furthermore, HTM has been proven to improve the performance of applications in many fields, such as network modelling (Chakravarthy et al., 2012) , drug discovery (Mitri et al., 2017), vision systems (Afeefa & Thulasidharan, 2018), smart homes (Alshammari et al., 2019), digital forensic, market analysis, business modelling, language processing, identification, and geospatial tracking (Cui et al., 2016; Hawkins and Ahmad, 2016; Hawkins and George, 2006; Hawkins and Blakeslee, 2004). Hence, it is reasonable to expect that HTM can be adapted to a smart transport application and improve the capabilities of these systems. Thus, it is worthy to investigate the use of HTM in this thesis.

This research will implement a multi-level anomaly detection framework using the HTM algorithm. The framework also accepts a feedback connection from a prediction model, which can help to improve traffic anomaly detection in smart transport. The outcome of this thesis is expected to be useful to other researchers in the smart transport field seeking to develop new solutions to traffic detection problems..

1.2 Project Aim

This thesis aims to study the requirements of a smart transport traffic detection application under the IoT paradigm to propose a novel framework based on HTM theory to detect anomalies in road traffic data as part of a smart transport management application.

The project first investigates smart transport and identifies its requirements and open issues. After that, it examines the HTM working theory to be applied and adopted for the proposed framework.

1.3 Project Objectives

The following objectives were satisfied to achieve the aim of this project:

1. Review the existing IoT technologies, applications, and challenges.
2. Critically review smart transport application domains to identify their functional and technical requirements and datasets.
3. Review and analyse existing datasets to address traffic detection in smart transport.
4. Review existing machine learning algorithm solutions that are used to detect anomalies in smart transport.
5. Investigate HTM theory and the Cortical Learning Algorithm (CLA), the implementation of the HTM, to investigate the use of HTM theory as practical anomaly detection algorithms for smart transport.
6. Identify existing open dataset(s) that represent smart transport scenarios and are suitable for use in testing and evaluating the proposed framework.
7. Propose a novel multi-level anomaly detection framework based on the HTM theory to tackle traffic problems in the smart transport domain.
8. Implement and evaluate the proposed framework using the latest version of the Numenta Platform for Intelligent Computing (NuPIC) platform.
9. Improve the performance of the proposed novel multi-level anomaly detection framework by introducing a feedback loop to the framework to enhance

anomaly detection.

10. Test and critically evaluate the improved framework with the feedback loop against conventional anomaly detection machine learning methods.

1.4 Contribution to Knowledge

The novel contributions to knowledge of this thesis are:

1. The main contribution to knowledge is a novel multi-level anomaly detection framework based on HTM theory to detect anomalies in traffic. On the first level, anomalies are detected based on the congestion level in the training datasets. On the next level, the combined anomalies are fed to a prediction model to learn to predict anomalies in the future.

2. The second major contribution to knowledge is to improve the learning capability of the proposed multi-level framework by introducing a feedback loop that can learn from patterns of data to detect anomalies in the context of smart transport traffic management.

3. A secondary contribution of this work is a pre-processed subset of the Motorway Incident Detection and Automatic Signalling (MIDAS) traffic data used in the proposed framework. This contribution required a comprehensive review of available datasets for use in evaluating traffic congestion machine learning-based solutions and identifying the suitable set.

4. Another secondary contribution to knowledge is to optimise the model parameters to identify the best parameter settings for the traffic congestion managements in smart transport.

5. Another secondary contribution to knowledge is the performance evaluation of the proposed framework and some conventional anomaly detecting machine learning methods using accuracy, recall, precision, and F-measure as performance metrics.

1.5 Research Questions

The work reported in this thesis attempts to answer the following research questions:

- Is the HTM theory suitable for the smart transport application?
- How can the features of the HTM theory, such as the hierarchal structure and online learning could be used to improve the performance of detecting anomalies?
- How is it possible to generate and prepare suitable data sets that can be applied to evaluate machine learning (ML) techniques and algorithms in smart transport scenarios?

1.6 Scope of the Research

This research will implement a novel framework based on the HTM theory. This framework will attempt to improve traffic congestion detection, which is a requirement of smart transport. This thesis focuses on the application of HTM machine learning algorithm in detecting abnormalities in real-time traffic data.

The description of the general scope of proposed work in this thesis appears in figure 1-1. As the figure shows, the IoT paradigm applies to many applications, such as smart homes and health care In this thesis, the research focused on smart transport as part of the IoT applications domain, including challenges and issues, such as reducing costs, accidents, and traffic congestion control.

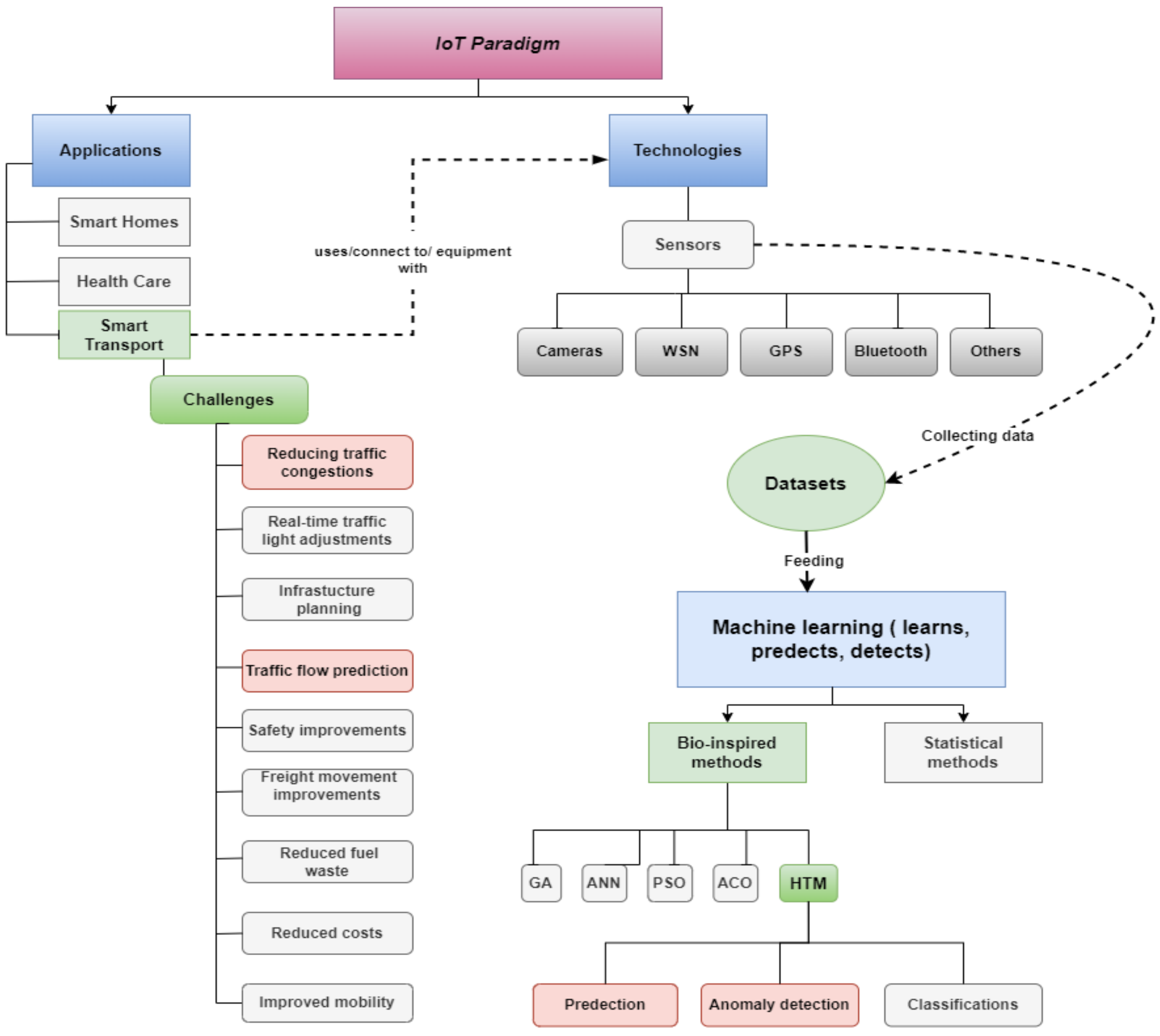


Figure 1-1: General scope of the thesis.

Traffic congestion is one of the open issues that affect people's lives in many ways, causing economic and health problems worldwide, particularly indeveloping countries (Zaki & Ali-Eldin, 2018). As population growth rates increase rapidly, road networks become increasingly unable to cope with traffic demand, leading to increases in queuing at junctions, travel delays, fuel consumption, and air pollution (Mallik, 2014). The focus of this thesis is therefore on detecting abnormalities from traffic data to enhance detecting congestion as early as possible.

As illustrated in Figure 1-1, transport infrastructure is now equipped with IoT sensing technologies to collect information about vehicle traffic and road conditions. The data are then fed into machine learning models. Studies on the use of machine learning techniques with smart transport have included statistical and bio-inspired methods. However, statistical methods were beyond the scope of this thesis. Hence, this research focused on the HTM technique as one of the bio-inspired methods, as described in Chapter 3 of this thesis.

This research examined the smart transport application domain under the IoT paradigm. Current issues and gaps in the state of the knowledge are reviewed in Chapter 2. Datasets generated from IoT sensing technologies are examined in Chapter 3. The datasets were fed into machine learning models to perform prediction and anomaly detection tasks. The HTM technique, a bio-inspired machine learning method, was the focus of the research effort.

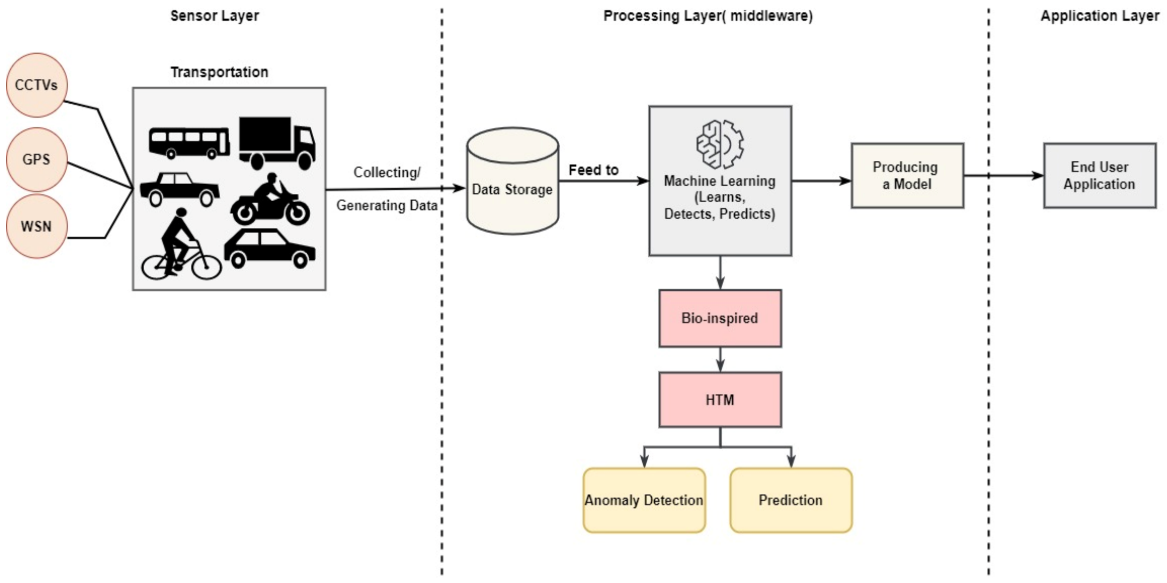


Figure 1-2: Details of the scope.

The data that will be used are collected from road sensors such as Geographic Point System (GPS) sensors.

Figure 1-2 illustrates the details of the scope of the research presented in this thesis in the context-aware concept layers. As mentioned previously in this section, IoT sensor technologies such as GPS generate huge amounts of data in the sensor layers. The data then are saved and moved to the processing layer, as shown in Figure 1-2.

This project also examined available datasets and how to use them. It should be noted that how sensors work and generate data were beyond the scope of this research. In addition, to the best of the researcher's knowledge, no previous research has identified datasets suitable for use in evaluating anomaly detection scenarios. Hence, this thesis included a comprehensive review of relevant datasets that could be used to evaluate anomaly detection scenarios.

Furthermore, there is a demand for a machine learning algorithm with a level of intelligence to do tracking, modelling, and prediction tasks. Hence, the work at hand will investigate machine learning-based solutions related to anomaly detection and prediction approaches.

Figure 1-2 also highlights the deployment of these techniques in the middleware layer. The proposed framework is designed and implemented in this layer. The final application will be used by targeted users in the application layer.

HTM has promising potential for any application that involves anomaly detection, object classification, and prediction. The review and analysis of smart transport requirements revealed that traffic congestion was an issues that must be addressed. Thus, this thesis proposes a novel framework based on HTM theory to detect anomalies in road traffic as part of a smart transport management application.

1.7 Research Approach

Methodological considerations and decision-making are critical to realise the research goal and objectives informed by a critical review of the empirical literature. Research methods are the data collection tools and techniques that are used in empirical research studies in various disciplines (Bryman, 2016). Many research methodologies are applied by researchers depending on the nature of the problem to solve (Davies & Hughes, 2014). One of the research methodologies used in this thesis is the research onion model.

The research onion model is a concept that Saunders et al. (2019) developed, and it explores the diverse decisions that researchers make when constructing a research methodology in research work. The concept demonstrates that when working, researchers start with high-ranked choices and then begin working inwards towards the least philosophical and tactical concepts (Saunders et al., 2019).

This research model can be used for many research projects (Saunders et al., (2019). It describes the phases of a research project, beginning with the research philosophy and ending with strategies and methods for data collection.

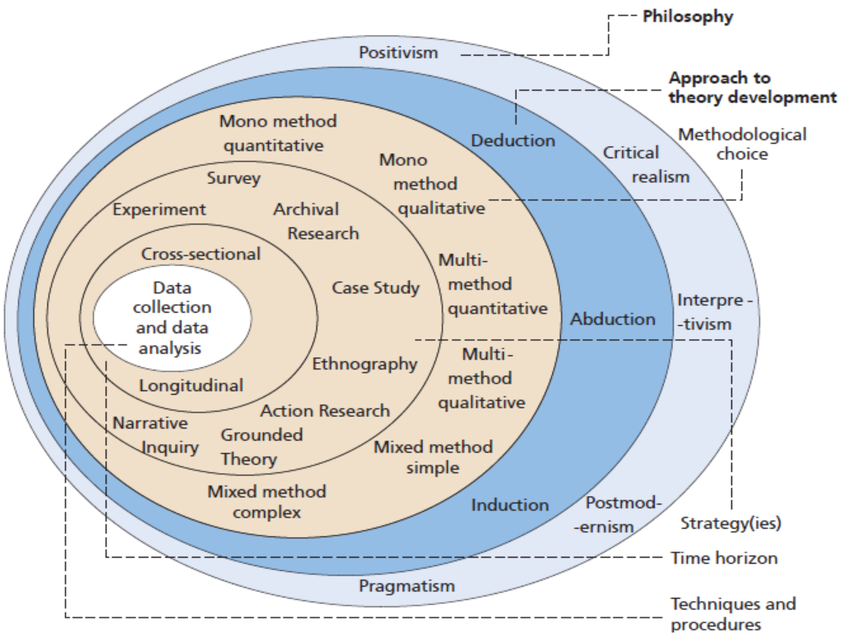


Figure 1-3: The research onion model adopted from (Saunders et al., 2019).

Figure 1-3 presents the research onion model. It comprises six layers concerned with determining the research philosophy that encompasses the origin, nature, and evolution of knowledge (Saunders et al., 2019). These layers are philosophy, approaches, choices, strategy, time horizon, and the final layer is techniques and procedures.

The research onion model's outermost layer is the research philosophy**.** This layer concerns the philosophical underpinnings of all knowledge pursuits, which is the first point of consideration in a research methodology.

It is usually the first and most important step before constructing a research methodology. This layer comprises multiple mechanisms: positivism, interpretivism, and pragmatism, which work within diverse ontological and epistemological hypotheses. The positivism mechanism hypothesises that knowledge thrives outside of what is being experimented with; this implies that the experimentation under study can be executed objectively and does not necessarily need to incorporate individual assumptions (Saunders et al., 2019). In addition, positivism argues that there is always one reality and connotation across themes. Hence, knowledge is only gained by empirical research and observation. Therefore, selecting suitable methods or mechanisms in this layer is appropriate because they are the cornerstone of the research and mirror the researcher's ontological and epistemological hypothesis. As a result, they aid in determining the first action that needs to be identified and the values to be adopted based on the type of research.

In establishing a research philosophy, positivism is frequently used in applied science. Interpretivist is often considered the polar opposite. The interpretivism perspective on reality posits that numerous realities rely on observers' interpretations. Furthermore, these realities are temporal and context-dependent (De Villiers, 2005). Based on these facts, this thesis adopts the positivism philosophy, which holds that science is the only path to discover the truth. Positivist studies believe that the research is objective and accomplished with the least amount of interaction with research participants.

The onion model's second layer, which is research approaches associates itself with suitable approaches under each philosophical perspective on reality. In short, approaches are the methods that a researcher is supposed to use in their research, inductive and deductive (Young et al., 2020). Henceforth it is always important to select the research approach to be followed, for it helps in understanding the decisions to be made. The deductive method involves formulating a hypothesis based on a pre-existing theory.

This hypothesis is either supported or rejected based on gathered evidence. As a result, the logical technique progresses from general to specific. It is used together with quantitative research, and they are used together because they all start with a hypothesis that has to be tested (Young et al., 2020).

In contrast, the inductive technique starts from being specific and moves towards being general. It begins with a collection of observations. Then, it attempts to uncover patterns in these observations to make hypotheses, which can eventually develop into a theory.

Choosing between inductive and deductive methods is important because it aids a researcher in determining the most appropriate approach to carrying out a study by focusing attention on the type of experiment that needs to be undertaken and particularly whether the research methodology needs to be derived from existing concepts (Young et al., 2020).

Therefore, the deductive approach was chosen for this research study. Since the research involved formulating hypotheses based on HTM, the deductive approach was better suited to testing the hypotheses considered during the research phases.

The methodological choices represent the third layer in the research onion model, which includes two types of methods to choose from while conducting a study. These types are qualitative and quantitative. There is another form of research that combines the use of both types of research methods, and it is known as a mixed-methods approach.

Qualitative researchers are primarily concerned with the process and meaning (Atieno, 2009). Moreover, qualitative research methods are subjective and provide textual information to develop theoretical insights (Davies & Hughes, 2014). Qualitative research methods are predominantly used in social sciences and management research. Interviews, content analysis, and participant observations are some of the tools and techniques employed to collect qualitative data.

In contrast, quantitative research methods entail objective measurements and the statistical analysis of data collected in a research study (McQueen & Knussen, 2002). Furthermore, quantitative methods can potentially give the general surface structure of the topic on a large scale within a research study. Researchers use quantitative methods when the research design is fixed before the data collection process starts (Robson and McCartan, 2016). Quantitative research methods assume that everything in the social world is described or measured with numbers. They allow large data sets to be collected and analysed logically (McQueen and Knussen, 2002). Quantitative research methods are more appropriate in natural science or laboratory research work and can test a hypothesis in experimental studies (Hopkins, 2008).

This research considered three reasons to choose the quantitative data gathering strategy. First, a quantitative research method is typically linked with the deductive approach and positivist philosophy. Second, the research relied on an enormous volume of data from various types of traffic sensors in the experiment stage. The experiments measured the effectiveness of the HTM machine learning framework at identifying anomalous patterns using these data. Finally, the hypothesis is tested using statistical metrics such as Accuracy, Recall, F-measure, and Precision.

The fourth layer of the research onion model, called the strategies. It involves a step-by-step plan of action that directs thoughts and efforts, enabling a researcher to conduct research systematically. The research is usually carried out using various approaches such as experimental research, action research case studies, grounded theory, surveys, and archival and narrative inquiry (Saunders et al., 2019). The benefits of these strategies are their flexibility in assisting with data collection and their capability to detect relationships among situations.

Experimental research involves manipulating a variable to observe whether it produces a change in another variable. This approach ascertains relationships among variables and is centred on scrutinising existing concepts. In contrast, action research involves learning by waiting for an action to occur and is carried out in practical situations. It is usually intended to make researchers aware of a particular challenge that occurs in a realistic world. Another strategy is a case study, which is usually a comprehensive analysis of a single topic (Saunders et al., 2019).

As was stated in this section, this research tested hypotheses based on HTM; therefore, the deductive approach was judged to be better for testing these hypotheses. Hence, the experimental strategy was chosen from among the available strategies identified by the onion model to test the hypotheses proposed in this thesis. Several experiments were conducted to attempt to answer the formulated research questions.

The Time Horizon is the fifth inner layer of the research onion model, and explains how much data researchers are planning to gather over a given period of time (Saunders et al., 2019). The two main time horizon options are cross-sectional and longitudinal time horizons. A longitudinal time horizon is mainly used when examining changes over time. In longitudinal studies, researchers normally examine data and collect variables without influencing them. This approach is appropriate for identifying patterns of changes and actions over a given time period. A limitation of this approach is the inability to predict changes that occur in data over time.

A cross-sectional time horizon is used to collect data at a given time to detect changes occurring with the data. Many researchers prefer a cross-sectional study time horizon because it is fast and inexpensive compared to a longitudinal horizon, which takes considerable time to finalise (Birks & Mills, 2015). A cross-sectional time horizon was chosen for this work as the experimental settings were changed over time to test the feasibility of improving the proposed framework.

Techniques and the procedures are sixth layer of the onion model layers. Here, researchers consider actual practicalities in their study to decide on techniques. Researchers decide on the data types they will collect and the methods they will use, such as questionnaires, surveys, or observations. In this layer, the researchers choose the data analysis method best suited to answering their research questions and decide how they will sample the data or study participants, if any (Saunders et al., 2019). The data analysis techniques and procedures selected here help the researcher gain a deeper insight into the problem being studied and assist in mining scientific and logical explanations and conclusions that help answer the research questions and the problem under study. They assist in developing new frontiers of information because of their ability to aid researchers in gaining knowledge when analysing scientific information.

As part of this thesis's contribution to knowledge in the field, the datasets used play an important part in testing the proposed framework. The techniques and procedures used were related to the quantitative data analysis conducted to measure the effectiveness of a HTM machine learning framework. Furthermore, the methods and the procedures used fulfilled the third objective of this thesis research, which was to review and identify suitable datasets, as discussed in more detail in Chapter 4.

The HTM theory and the associated hypothesis that the machine learning framework can effectively identify the anomalies guide the experimentation and development of the machine learning framework. As a result, this study takes a positivist approach that uses the deductive research technique. It starts with the HTM theory and employs quantitative research methods to demonstrate HTM's applicability in detecting anomalies in traffic data. The major stages of this study appear below.

1. ***Secondary Research Phase:*** This phase involves the related theory to get a deeper understanding of all concepts and relevant work that developed the framework proposed in this thesis. It started with a review of the literature. The literature review provides an overview of related work that will be an aid to develop and implement the designed framework for determining abnormalities in data streams to detect congestion before a crucial event happens. The research concentrates on AI-related sources and contains major related works from disciplines like anomaly detection, machine learning, data mining, IoT, smart transport monitoring, and traffic congestion.

Relevant literature was identified using many keywords such as AI, data mining, hierarchical temporal memory, anomaly detection in road traffic monitoring scenario, traffic congestion, highways traffic detection, machine learning algorithms, and traffic monitoring. Queries addressed the *Google Scholar, SpringerLink, IEEEXplore Digital Library, ScienceDirect, and ACM Digital Library*. The investigation of relevant research also included the reference list shown in publications from reputable journals and conferences. Furthermore, a dedicated research project has selected a dataset representing real traffic records to train and evaluate the suggested model. This data is analysed to determine the requirements for detecting the anomaly algorithms.

Additional research on the HTM theory is carried out to get a deeper understanding of the HTM's hierarchical organization, accompanied by a study of the HTM's main principles, such as regions, hierarchy, time, and sparse distributed representation. The results of this phase are analysed to determine the HTM theory's stability and capabilities in recognizing abnormal data points within the area of smart transport congestion detection.

2. ***Primary Research Phase:*** the primary research focuses on studying the data gathered during the literature review phase. This phase entails a more methodical and theoretical examination of the theory that underpins the investigation. The findings of this phase serve as the foundation for the suggested solution's design, implementation, and the eventual process of performance evaluation.

For the scenarios related to congestion detection in the smart transport domain, an innovative and adaptive framework is presented, designed, and deployed based on the HTM theory. After that, the examined datasets of real traffic situations are pre-processed and prepared for testing and evaluating the proposed framework. Finally, the experiments are performed to assess and compare the proposed framework's performance to various conventional methods.

1.8 Structure of the Thesis

This thesis is organised as shown in Figure 1-4 into the following seven chapters:

Chapter 1: presents the problem definition, project's aim, objectives, scope of the research, research questions, and main contributions.

Chapter 2: reviews the IoT literature and focuses on smart transportation and machine intelligence algorithms that cover related work and open research gap.

Chapter 3: presents the Hierarchical Temporal Memory (HTM) theory and its components, encoders, spatial pooler, temporal memory, and the Cortical Learning Algorithm (CLA), which is the implementation of the HTM theory.

Chapter 4: presents the datasets for smart transport problems. This chapter covers related real datasets and synthetic data in the literature, and it describes the criteria used to choose datasets in this study.

Chapter 5: mainly focus on the first main contribution to knowledge. It introduces first the design and requirements of a machine-based smart transport application. Then, the second part of this chapter describes the proposed Multi-Level anomaly detection framework.

Chapter 6: covers the implementation and testing of the improved Multi-Level anomaly detection framework that was implemented in Chapter 5 with prediction feedback. Next, it evaluates the proposed framework against current conventional machine learning methods and the standard HTM.

Chapter 7: presents a summary, conclusions drawn from the research results, and suggestions for future work.

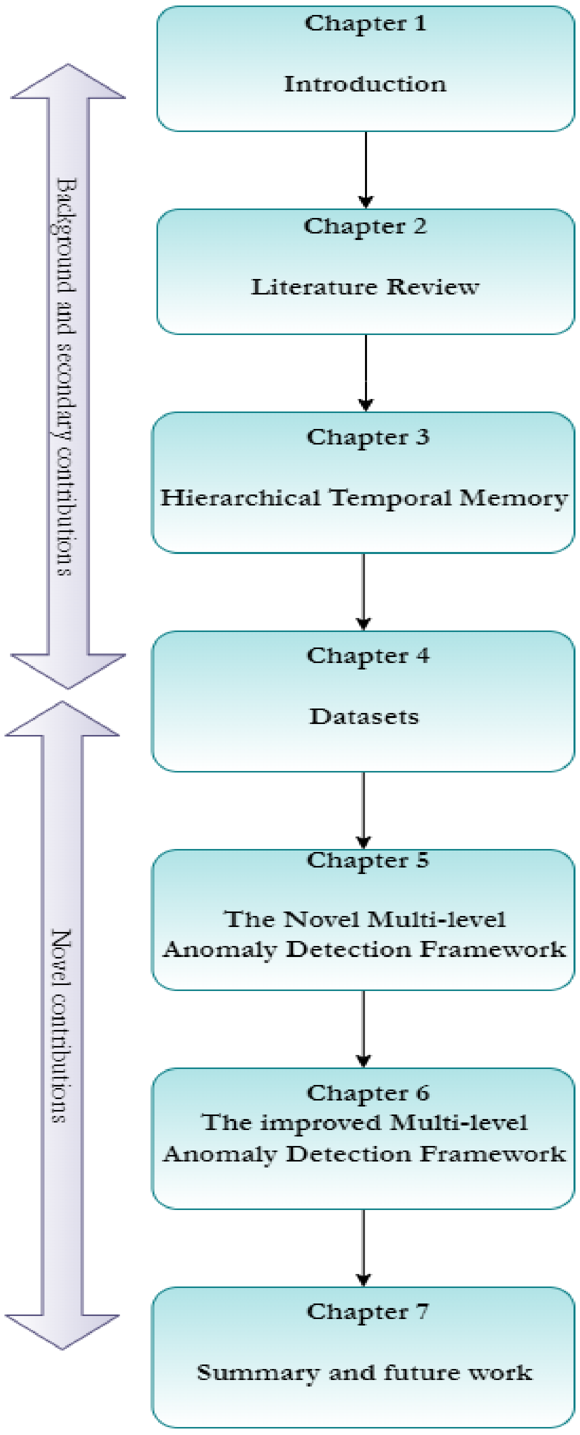

Chapter 1
Introduction
Chapter 2
Literature Review
Chapter 3
Hierarchical Temporal Memory
Chapter 4
Datasets
Chapter 5
The Novel Multi-level Anomaly Detection Framework
Chapter 6
The improved Multi-level Anomaly Detection Framework
Chapter 7
Summary and future work
Background and secondary contributions
Novel contributions

Figure 1-4: Thesis structure.

Chapter 2 Review of Literature

2.1 Introduction

One of the most critical concerns in the transport sector is how to address traffic congestion. Many solutions have been proposed to address this problem (Del Ser et al., 2020). However, there is a need for more sophisticated solutions that reflect the changes in the road infrastructure. With the aid of the latest technologies, smart transport has become more desirable and offers potential solutions to make roads less congested. This benefits the environment by reducing fuel consumption and benefits travellers by reducing travel times and costs.

As mentioned in the background section in Chapter 1, various machine learning tools such as ANN (Figueiras et al., 2018) have been employed to examine traffic congestion. Nevertheless, the foreseen benefits of machine learning-based solutions and the number of publications in this research area remain limited and must be fully investigated.

This chapter presents a comprehensive review of related published literature covered on smart transport solutions to traffic congestion. This chapter first reviews the Internet of Things (IoT), including its major issues. An intelligent transport system is then reviewed. Machine learning algorithms are reviewed to describe the current state of research and identify the gap addressed by this thesis.

This chapter is organised as follows. Section 2.2 explores the Artificial Intelligence (AI) field, including the importance of AI, the goals of AI, and the computational tools for AI. The section also explores application examples that demonstrate new research developments. Section 2.3 details the definition, history, and applications of the Internet of Things. This section also describes new ways of generating data, especially for intelligent transport systems, which open new opportunities for testing and using IoT devices in traffic management systems. Section 2.4 reviews machine learning models and their assessment. Section 2.5 assesses machine learning approaches in the context of smart transportation, particularly anomaly detection. Section 2.6 investigates the role of machine learning in traffic congestion management approaches. Section 2.7 reviews intelligent transport and discusses the impact of traffic congestion in the UK and worldwide. The causes of traffic congestion, traffic modelling, and anomaly detection are reviewed in this section. The section also discusses the use of big data for intelligent transport. Section 2.9 reflects on the literature review findings, identifies the research gap addressed by this thesis, and summarises the highlights of the chapter.

2.2 Artificial Intelligence

Artificial Intelligence (AI) is the simulation of human intelligence in machines by programming them to think like humans and copy their actions.

According to Haenlein & Kaplan (2019), the history of artificial intelligence can be traced back to the 1950s, when it was established as an academic discipline. It is further indicated that artificial intelligence can be classified into analytical, human-inspired, and humanized AI. Haenlein & Kaplan (2019) offer a chronological history of the developments pertaining to AI. The authors opine that although it is difficult to pinpoint the root of AI, it can probably be traced back to 1940s.

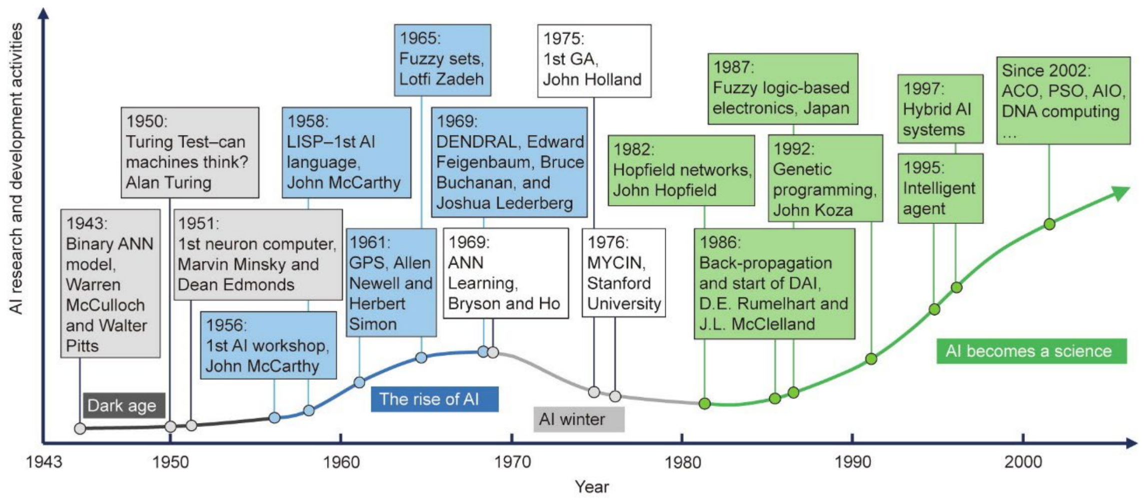


Figure 2-1: The History of AI adopted from (Wang, 2019a).

AI term was first introduced in 1956 by John McCarthy as *“the science and engineering of making intelligent machines”* (McCarthy et al., 2006). Figure 2-1 Illustrates the history of the AI.

Wang (2019) asserts that there is no widely accepted definition of an AI, and therefore the term is used to describe many different aspects. This, therefore, indicates there are varying definitions of artificial intelligence. According to Hilb (2020), artificial intelligence is the activity devoted to making machines intelligent, making them function appropriately, and foresight in their environment see Figure 2-2 and section 2.4 which explained the Machine Learning part. The main characteristic of AI is that it can rationalize and act in a given way to achieve a specific goal. Artificial intelligence, therefore, indicates a machine that exhibits characters associated with the human mind, like learning.

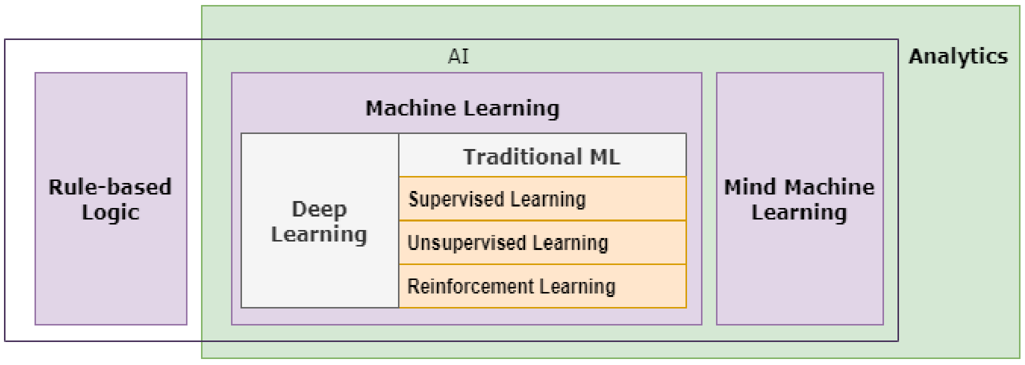


Figure 2-2: Artificial Intelligent adapted from (Hilb, 2020).

2.2.1 Goals of using Artificial Intelligence

The main reason to use AI is to create technology that makes computers and machines work intelligently. Artificial intelligence is used to solve problems and puzzles and address uncertainty and incomplete information in making logical deductions. Computers store large volumes of memory that AI uses to process information and solve complex arithmetic problems (Vinuesa et al., 2020). Artificial intelligence is applied to use machines to solve various problems that a person would otherwise need to solve. Computers and machines use AI to think like human beings and do things that require human intelligence, such as learning and problem-solving. Artificial intelligence is used to improve decision-making, especially in big data analytics.

Organisations use artificially intelligent machines to analyse data to make decisions and future plans (Mathur & Burns, 2019). The other goal of using artificial intelligence is to optimise business processes by increasing productivity and operational efficiencies. Artificial intelligence is used by banks, insurance companies, and financial technology FinTech and insurance technology InsurTech companies. Personal digital assistants and chat-bots are used by banks, insurers, and other organisations to respond quickly to the concerns of users (Mathur & Burns, 2019). Artificial intelligence is also used in Automated Teller Machines (ATMs) and in online banking to ensure that the right transactions, are applied to accounts correctly based on the account status and transaction history so that any unexpected or incorrect transactions can be identified promptly(Kaur et al., 2020). The other goal of artificial intelligence use is to develop systems capable of recognising, processing, and simulating human effects.

2.2.2 Computational Tools for Artificial Intelligence

Computational intelligence is the theory, design, application, and development of linguistically motivated computational programs. Computational tools used in artificial intelligence include Scikit Learn. According to Pike (2019), it is essential to understand three features of a computational ecosystem to view AI as a suite of computational tools and not "intelligence". These features are that there is no universal algorithm, and employing AI is not rewriting the selected tool for each new problem being analysed. The third feature is a collaborative environment. Pike (2019) asserts that Scikit Learn offers highly optimized libraries for machine algorithms. Scikit Learn is a Machine Learning library that underpins various administered and unsupervised learning calculations. This tool expands on two critical libraries of NumPy and SciPy and Python. Scikit Learn performs many calculations for AI and data mining assignments, such as bunching, relapse, and order. Tensor Flow is another example of a computational tool; it is an artificial intelligence library that utilizes an arrangement of multi-layer hubs that enable it to send counterfeit neural systems.

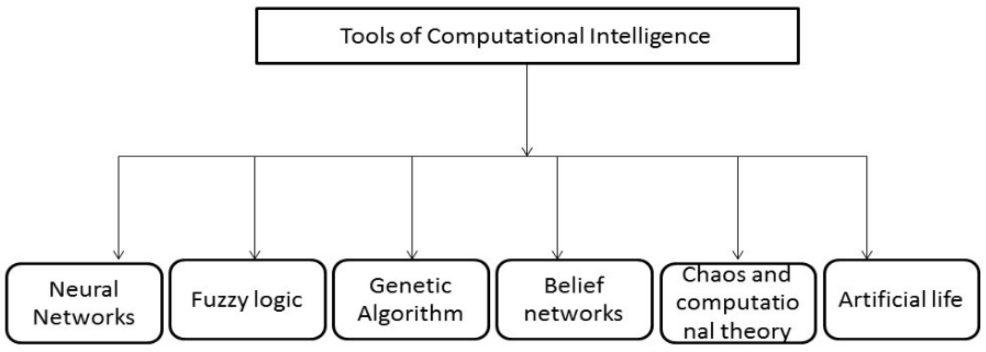


Figure 2-3: Computational Tool for Intelligent adopted from (Raj, 2019).

When a programmer composes a Python program, Tensor Flow helps arrange and keep running on Central Processing Unit (CPU) or Graphics Processing Unit (GPU). This means one is not supposed to compose at the C++ for them to keep running on GPU. The other computational tools as shown in Figure 2-3 are neural networks, fuzzy logic, genetic algorithm, and belief network. Raj(2019) asserts that other computational tools or tools of computational intelligence are artificial life and chaos and computational theory. It is further indicated that computational tools with fault tolerance are artificial neural networks and fuzzy logic, making them suitable for use in modern society because they have an optimal solution for the problems.

2.2.3 Applications of Artificial Intelligence

Artificial intelligence is applied in almost all aspects of the contemporary world. Abduljabbar et al. (2019) assert that the rapid pace of development in AI is offering unprecedented opportunities to enhance the performance of different businesses and industries. Artificial intelligence is used in various sectors such as marketing, banking, finance, and gaming. AI is also used in agriculture, health, and space exploration, among other areas. Li et al. (2017) assert that artificial intelligence technology is used in the manufacturing industry to enhance productivity. Industries and manufacturers use computers and robots that are programmed to perform specific or multiple tasks. Banks, insurance, and supermarkets use chat-bots to interact with customers. AI is applied in machine learning to solve problems and analyse data. Institutions use artificial to analyse large data to make critical decisions like marketing. Artificial intelligence is used to list products online where consumers can buy them. By using the Internet of things, a consumer can visit an e-commerce store and select a product. It is also possible for a person to seek guidance or recommendation of products or services online from chat-bots.

Artificial intelligence recommends products and services based on the websites and information that a person had interacted with online previously. For example, when a person is searching for a cloth, car, or movie online, various products appear in form of advertisements. Artificial intelligence is applied in the agriculture sector, whereby companies use automation and robotics to find more efficient ways to protect crops from weeds. According to Eli-Chukwu (2019), the application of AI has been evident in the agricultural sector recently. It is further indicated that AI is applied in agriculture concerning soil, crop, diseases, and pest management. For example, AI technology is used to harvest cucumber by robots that include autonomous vehicles, manipulators, and end-effector. In the healthcare sector, artificial intelligence is used in digital acquisition and computing infrastructure (Yu et al., 2018).

2.3 Internet of Things (IoT)

2.3.1 Definition

Internet of Things (IoT) refers to a network of physical objects emended with sensors, software, and other technologies to link and exchange data over the Internet. According to Srinivasan et al. (2019), the IoT exemplifies that the network of devices can acquire and share information. In essence, this means that devices or things are connected to share information using the Internet. Srinivasan et al. (2019) further assert that the network devices use internet protocol or IP to communicate with one another. Figure 2-4 provides an illustration concept of the IoT as the interconnection between people and things (Firouzi et al., 2020).

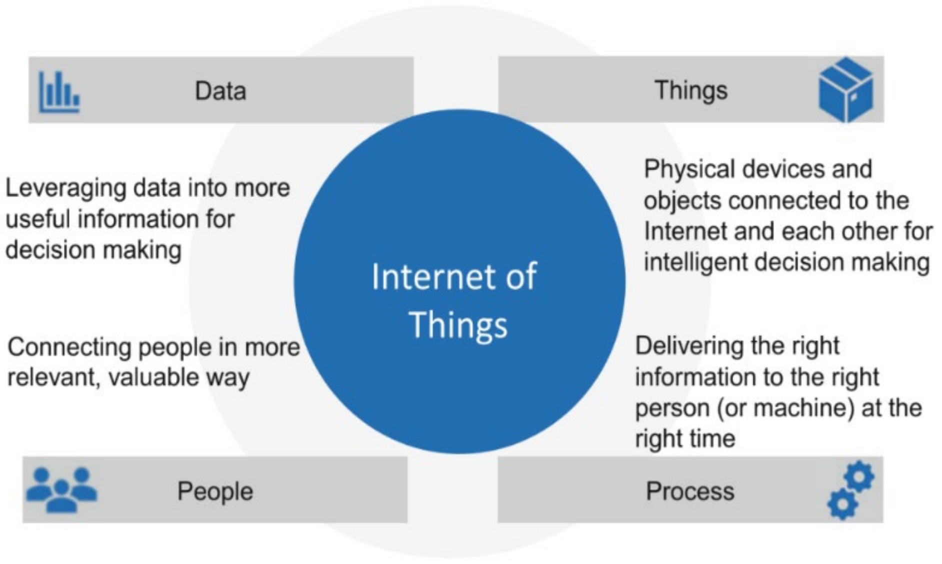

Data
Things
Leveraging data into more useful information for decision making
Internet of Things
Physical devices and objects connected to the Internet and each other for intelligent decision making
Connecting people in more relevant, valuable way
Delivering the right information to the right person (or machine) at the right time
People
Process

Figure 2-4: The IoT definition and visions adopted from (Firouzi et al., 2020).

2.3.2 History Internet of Things (IoT)

Kevin Ashton invented the term Internet of Things in 1999 during his presentation to Proctor and Gamble about incorporating RFID (Radio Frequency IDentification) tags within the supply chain. According to Chin et al. (2019), the term IoT was coined by Kevin, who argued that incorporating RFID tags empower computers with their own means of gathering information for them to see, hear, and smell the environment around them. The history of IoT can be traced to the emergence of the electromagnetic telegraph in the 1830s and the rise of connectivity and computers. The creation of modems and communication between electronic devices laid the foundation for the development of IoT. The first mobile phone was developed in the 1970s, followed by the creation of Sim-card that allowed devices to connect. Chin et al. (2019) outlined different phases of the history of the development of IoT. Between 2005 and 2008, the IoT was in its infancy. This period is referred to as "the devices and connectivity period", whereby networking, communication, and identities were the key indicators. Cloud computing was in its infancy, and this term did not exist then. The objective of the Internet back then was to transform everyday objects into embedded computers and to provide the object with identity. These things were then to be connected to the Internet.

The second phase came between 2009 and 2011 and was characterized as the machine-to-machine period. In this period, there was a gradual improvement to support functions of the Internet. These include the creation of networks and standards and machine-to-machine communication. There was a massive shift from hardware connectivity to software connectivity, data, information, and services. Between 2012 and 2014 came the third phase, known as the HCI period, which was characterized by object identification, end-user programming, and wireless connectivity. The period from 2015 to date is referred to as "the smart period" and is characterized by increased devices using the Internet of things.

2.3.3 Goals of using IoT

The main goal of the Internet of things is to connect physical devices like TV, sensors, and watches to the Internet to create a network to communicate with each other. The connectivity of these devices is to enables them to make decisions intelligently. The other goal is to change how people interact with things and make life simpler. For example, a person using a CCTV camera connected to a phone or laptop can interact with their co-workers both at home and office (Muthusenthil & Kim, 2018). IoT also aims to enhance the security of things like computer devices, cars, and other properties (Ali & Asif, 2018). For instance, trackers fitted in vehicles or put on things like pets' collars help trace one's car or pet.

There is a generation of the high value of data by devices connected to the Internet. Therefore, analysing these data helps in decision making or intelligence machines. Internet of things is used to coordinate devices for them to function as a team. For example, robots interconnected with other machines in an industry work effectively to deliver the services they are programmed to deliver. The other goal of using IoT is to allow seamless communication between people, processes, and things at home and at work (Pan & McElhannon, 2018). Digital systems are used in recording, monitoring, and adjusting each interaction with connected things.

2.3.4 Internet of Things Applications

Internet of Things is applied in a wide range of appliances and things in the contemporary world to enhance communication. Elmustafa & Kamal (2017) opine that IoT has gained great attention from researchers because it is an important technology that allows communication between objects, machines, and everything together with people. IoT is used in smart homes where home appliances and products ranging from cars, fridges, CCTV, and TV are connected to communicate with each other. In such homes, a person can access security of the house, watch TV, switch on or off light from the comfort of their office using smartphones.

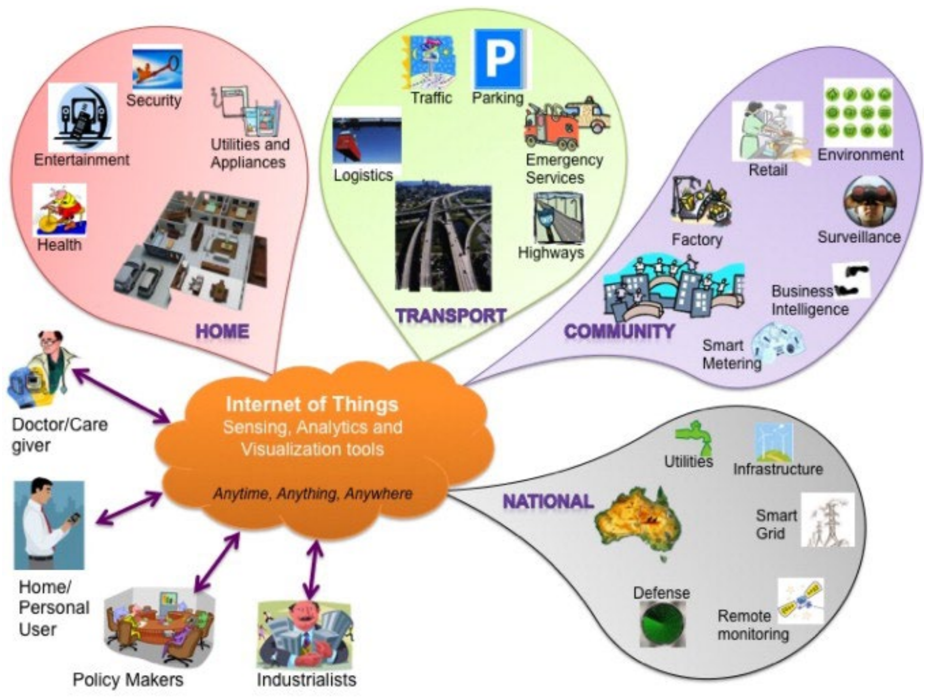


Figure 2-5: IoT application domains adopted from (Gubbi et al., 2013).

The other application of the Internet of Things is wearable such as Smartwatch, virtual glasses, and fitness bands that monitor calorie expenditure and heartbeats. In the health sector, IoT is integrated into hospital beds with sensors to monitor the patient's movement. According to Asghari et al. (2019), IoT is considered an ecosystem containing smart objects equipped with sensors, networking, and processing technologies integrated and working together. Figure 2-5 describes Atzori et al. (2010) and Gubbi et al. (2013) presented the applications of IoT as: (1) Smart cities such as smart parking, smart lighting, traffic management. (2) Home automation such as smart management of energy and water usage, controlling home appliances. (3) Smart Transportation and logistics domains such as assisted driving, traffic management, smart parking, and augmented maps. (4) Smart environments including forest fire detection, air pollution, monitoring snow levels (Gubbi et al., 2013; Atzori et al., 2010).

2.4 Machine Learning

Machine learning is a branch of artificial intelligence whose purpose is to make intelligent systems do the work of humans by extracting information from various types of data. Machine learning is applicable to situations in which data do not present patterns that can be represented mathematically. Machine learning is used to learn the mathematical relations of patterns in the data. According to Alpaydın (2014) Machine Learning is the technique enables computers to find unseen insights without having been explicitly programmed where to search. In other words, computers are given the power to learn without being overtly programmed. There are many types of machine learning algorithms, as summarised below.

2.4.1 Types of Machine Learning

This section describes different types of machine learning algorithms. The concept underlying each type of machine learning, how each type is commonly applied, and the limitations of each type are summarised.

2.4.1.1 Supervised Learning

Supervised learning is learning from labelled data. A subset of the data is labelled as the training set, and a separate subset is labelled as the test set and is used to test the performance of the model. The algorithm generates a function that can predict the outcome or target variable. In other words, the function can map inputs to desired outputs, as shown in Figure 2-6.

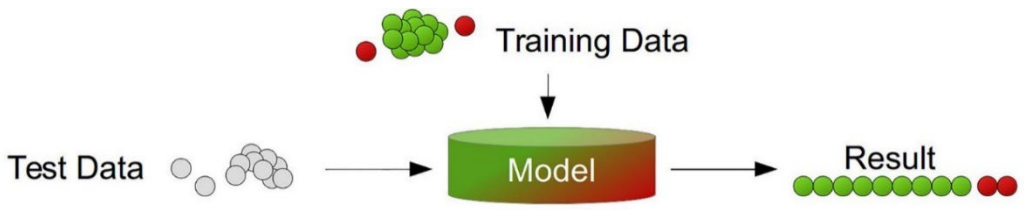


Figure 2-6: Supervised learning adopted from (Goldstein & Uchida, 2016).

The mathematical equations can be written using the mathematical notation in Rajeswari et al. (2018). For example, an algorithm can examine a dataset denoted as A in a dimensional input space associated with a labelled output space to find the function f that relates the two, represented in equation form as follows:

Let

$$A = \{(x_1, y_1), (x_2, y_2), \dots \dots \dots \dots \dots \dots . (x_n, y_n)\} \qquad 2.4\text{-}1$$

where xi is a specific input and y_i is the corresponding output. The equation data points are drawn from an unknown distribution S:

$$(x_i, y_i) \sim \mathrm{S} \qquad 2.4\text{-}2$$

The independent data points in dataset A are described as follows:

$$A = \{(x_1, y_1), (x_2, y_2), \dots \dots \dots \dots \dots . (x_n, y_n)\} \subseteq \mathrm{R}^d \times L \qquad 2.4\text{-}3$$

$$f: \mathrm{R}^d \rightarrow \mathrm{L} \qquad 2.4\text{-}4$$

such that for every new input/output pair (x,y) sampled from S there is a function f that:

$$f\ (x) \approx y \qquad 2.4\text{-}5$$

where n represents the dataset size, R^d is the feature dimensional space, x_i is the ith input, y_i *is the* ith output, and L is all possible labels in the space.

Among the many types of algorithms that can be classified as supervised learning algorithms (Du & Swamy, 2014; Ayodele, 2010) are decision trees, discussed in Section 2.4.2.2; regression discussed in Section 2.4.2.1; K- Nearest Neighbour (KNN) discussed in Section 2.4.2.4, neural networks discussed in Section 2.4.2.6, and finally the random forest discussed in Section 2.4.2.5.

Supervised learning can be classified into two sub classes: regression and classification. Regression is based on the already available features of the object of interest. Furthermore, the regression involves the prediction of other relevant features. Classification involves classifying objects into categories based on the available feature vectors of the objects (Campagner et al., 2021). Classification can include prediction: for example, predicting the cost of a building based on information about the area surrounding the building and the building price. Another example application of classification is determining whether to accept or reject a credit card application based on available client records. Classification has also been used to distinguish between spam and non-spam messages using message labelled information (Goldstein & Uchida, 2016).

The main weaknesses of supervised learning are that it can be very costly in some situations and can be impractical, as it depends on labelling, which may result in some cases in labels being misplaced or incorrectly defined (Campagner et al., 2021).

2.4.1.2 **Unsupervised Learning**

Unsupervised learning as the name implies is based on the analysis of available datasets. Unsupervised algorithms take unsupervised data as input and algorithms are applied for analysis without any human interaction. Such algorithms employ strategic techniques for the separation of objects of interest into different classes based on their feature vectors. No labelling or human interaction is employed for this purpose.

Unsupervised learning algorithms can be classified into the following two subclasses: association and clustering. Association algorithms find the appropriate relationships among the elements of particular datasets. Clustering algorithms, amalgamate data elements into particular sets based on their features (Domínguez-Barbero et al., 2020). There is no target of outcome variables to be predicted with these types of algorithms. In addition, clustering population in different groups can be achieved by using unsupervised algorithms.

There are many example for this sort of algorithms such as Apriori algorithm, K-means (Ayodele, 2010). The Hierarchical Temporal Memory is one of the most recent unsupervised learning, which described in Chapter 3.

Unlabeled Data
Unsupervised Algorithm
Result

Figure 2-7: Unsupervised learning adopted from (Goldstein & Uchida, 2016).

This is learning from unlabelled data as illustrated in Figure 2-7. It involves dividing data into groups based on their similarities. Such a collection of datasets denoted by D represented as:

$$D = \{(x_1), (x_2), \dots\dots\dots\dots\dots\dots\dots.\dots (x_n)\} \qquad 2.4\text{-}6$$

The aim of the unsupervised learning is to find similarities in datasets D to identify the function f as follows:

$$\forall x_i \in D \; \exists f(x_i) : \mathrm{T} \rightarrow \mathrm{y}_i \qquad 2.4\text{-}7$$

where: n represents the size of the dataset x_i is the i th input, y_i *is the* i th output space, and T is the collection of all possible similarities in the space.

Unsupervised learning is commonly used in application domains that involve identifying abnormalities or simply detecting anomalies. However, according to Marques et al.(2020) although unsupervised learning algorithms can be less costly to use compared to supervised learning algorithms, they are very sensitive to outliers.

2.4.1.3 **Semi-Supervised Learning**

This type of learning benefits from both supervised and unsupervised learning capabilities. This includes learning with both unlabelled and labelled information.

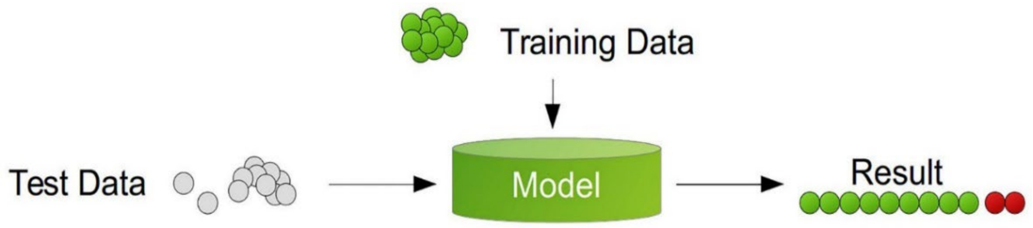


Figure 2-8: Semi-Supervised learning adopted from (Goldstein & Uchida, 2016).

Figure 2-8 illustrates the concept of semi-supervised learning, in which the data are labelled and unlabelled. When the amount of unlabelled data exceeds the amount of labelled data, both supervised and unsupervised learning techniques fail to impress. In such a scenario, unlabelled data are used to extract suitable information about objects of interest (van Engelen & Hoos, 2020). The mathematical representation shown in this section is an improvement on a formula presented by van Engelen & Hoos (2020) that describes semi-supervised learning as relaying information on dividing datasets into labelled and unlabelled data subsets. A denotes a given dataset represented in equations 2.4-9 through 2.4-11.

$$A = A_L \cup A_U \qquad 2.4\text{-}8$$

A_L represents labelled data and A_U represents unlabelled data.

$$A_L = \{(x_1, y_1), (x_2, y_2), \dots\dots\dots\dots\dots\dots\dots.\dots (x_n, y_n)\} \qquad 2.4\text{-}9$$

$$A_U = \{(x_1), (x_2), \dots\dots\dots\dots\dots\dots\dots.\dots (x_n)\} \qquad 2.4\text{-}10$$

There is a function denoted by f represents semi-supervised learning as follows:

$$f: p(y_i|x_i) \rightarrow \mathrm{L} \qquad 2.4\text{-}11$$

where n is the dataset size, x_i is the ith input, y_i *is the* ith labelled output, p is the probability of finding an associated label y for input x, and L is the learning outcome.

In semi-supervised learning, the amount of data to be predicted is always greater than the amount of labelled data, whereas, in supervised learning, the amount of data to be predicted would be less than the labelled data (Domínguez-Barbero et al., 2020).

Although semi-supervised learning can be less costly in the context of labelling and data preparation, since it falls between supervised and unsupervised learning, this can lead to less accurate performance in some cases (Sarker, 2021). An application of semi-supervised learning is the use of deep belief networks, in which some layers learn the structure of the information and other layers are used to define categories (Ayodele, 2010).

2.4.1.4 **Reinforcement Learning**

Reinforcement learning algorithms are used to make specific decisions. The key idea of reinforcement learning is that a machine learning algorithm is exposed to an environment so that the algorithm can train itself by trial and error. The algorithm includes the actions that software agents should consider in the environment. Reinforcement learning algorithms are helpful in learning how to achieve complex goals or capitalise on a certain dimension over various steps.

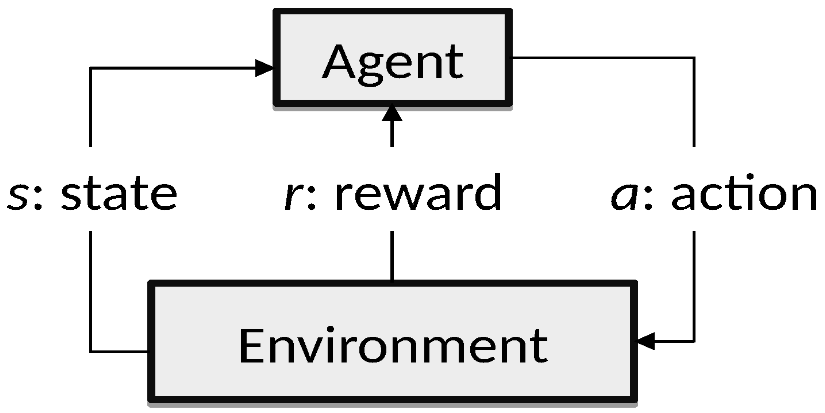


Figure 2-9: Reinforcement Learning adapted from (Domínguez-Barbero et al., 2020).

Figure 2-9 shows the basic structure of reinforcement learning algorithms. These algorithms tend to work in a sequential manner, with the optimal behaviour of the object evaluated at each step using its current state. The behaviour is then updated within a specific environment. The reinforcement algorithm, commonly known as the agent, predicts the feature of an object based on the causal values of the system, i.e., present and past. Based on the prediction, a reward is produced, which may act as a penalty if the analysis is not conducted carefully (Pugliese et al., 2021).

According to Krajna et al. (2022), reinforcement learning starts with a set of agents in a state called S that is mapped to an action or several actions called a to move to another state with a reword called R. The reinforcement model can be represented as shown in equation 2.4-13.

$$P: S_i \times a \times S_{t+1} \rightarrow \mathrm{R} \qquad 2.4\text{-}12$$

where S_i is the state at time i, and S_{i+1} is the new state after mapping with action a.

. The main advantage of reinforcement learning algorithms is that they can learn from experience and try to obtain the best knowledge available to make correct or accurate decisions (Domínguez-Barbero et al., 2020). However, reinforcement learning algorithms require time for deployment, which limits the resources of computers running reinforcement models, especially in tracking applications, where the actions in environments change over time (Krajna et al., 2022). Another limitation highlighted by Krajna et al. (2022) is that reinforcement learning depends on a specific environment; it is difficult to deploy a reinforcement learning algorithm for any environment other than the original one for which it was developed. This is especially true for applications involving change over time, such as transportation applications. A Markov decision process is a good example of a reinforcement learning algorithm (Ayodele, 2010).

2.4.1.5 **Other Types of Machine learning Algorithms**

The best-known types of machine learning algorithms are supervised, unsupervised, and semi-supervised learning algorithms. However, there are other types that are sometimes considered to be subtypes of these main ones or standalone types, depending on the application. An example is self-learning supervised learning, in which the algorithm learns its own inductive bias based on previous experience (Du & Swamy, 2014). This is the capacity to identify patterns, learn from the data, and become more intellectual over time. This is considered to be the future of AI. One application of self-learning supervised learning is self-driving cars. Self-supervised learning techniques are quickly becoming more popular because they eliminate the dependence on labelled data, a problem that is encountered with supervised, unsupervised, and semi-supervised learning techniques. Self-learning uses labelled and unlabelled datasets to train the model. Furthermore, the unlabelled data are initially trained using a pre-trained model and labelled data are directly fed to the supervised training model. In the supervised training model, the pretrained unlabelled data and labelled data are correlated for better classification of the object of interest. However, according to Bursztein (2022), self-learning supervised machine learning algorithms are not suitable for all types of applications, and sometimes they can be time-consuming and expensive to use.

Another example is feature learning, which is categorised as supervised feature learning or unsupervised feature learning. Supervised feature learning uses labelled input information and includes multi-layer perceptron and supervised neural networks (Sammut & Webb, 2011). On the other hand, unsupervised feature learning uses unlabelled data information and includes independent component analysis and various types of clustering. Feature learning is commonly used in image processing applications to classify pictures by selecting features based on certain specification (Längkvist et al., 2014). As the focus of this thesis research is on the transport domain, feature learning is outside the scope of this theis.

Sparse Dictionary Learning is another of machine learning algorithm that is sometimes considered a type of supervised learning. In sparse dictionary learning, data are presented as a sparse linear sequence of an overcomplete machine learning source set. This approach utilises a two-step iterative technique which includes either a clustering-based solution or convex relaxation (Tillmann, 2014). The basic aim of this technique is to identify the sparse representation of the linear combination of the data elements and the individual data elements, the individual data elements, which are also known as atoms.

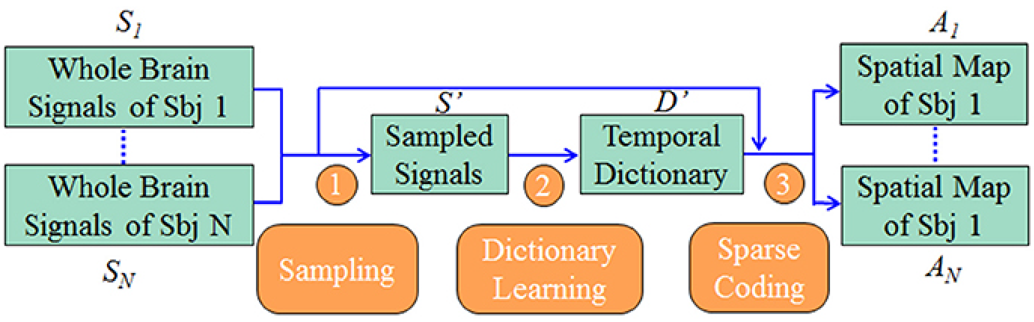


Figure 2-10: Sparse Dictionary Learning example adopted from (Ge et al., 2018).

Figure 2-10 illustrates the representation of data elements i.e., atoms as sparse representation matrices. The atoms are combined in a dictionary and their identification as sparse representation is of fundamental importance in terms of object identification and data classification. In the brain classification model, illustrated in the figure, the data elements detected from brain signals using electroencephalography are contained in matrix S. Matrix D contains the processed signal, while Matrix A contains the processed brain signals that are based on variations of the brain classification images obtained (Ge et al., 2018). After processing, the elements are aggregated before being fed into the dictionary learning routine for object classification. Using a sparse representation of elements and their subsequent modelling yields a very efficient yet very challenging tool. The atoms may end up in regenerative mode, which makes it very difficult to handle them cases in which the system starts to diverge from a solution (Sun et al., 2022).

Deep learning is an example of an artificial intelligence model that mimics how the human brain functions in terms of processing information and developing patterns that are utilised during decision-making. It is a type of machine learning that employs networks that can learn from unsupervised information that is unstructured or unlabelled. Therefore, deep learning is a method that teaches machines to do what humans do naturally (Janiesch et al., 2021). The main advantage of deep learning is that it classifies, clusters, and then processes a wide and unarranged set of data by employing a neural network. Since neural networks are trained to mimic human brain activity, they possess the ability to process large amounts of data using a set of predefined algorithms to identify a problem and solve it. This particular trait makes deep learning very useful and a topic of interest for many researchers (Shrestha & Mahmood, 2019). The most commonly used deep learning algorithms also known as deep neural networks are Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and Multi-layer Perception (MLP). Deep learning employs faster and much more efficient neural networks than other types of machine learning algorithms to combine three steps in the process of machine learning pre-processing, feature extraction, and selection into an efficient deep learning model (Alzubaidi et al., 2021). The layers of a deep learning model are represented as shown in equation 2.4-14.

$$z_j^i = {w_j^i}^T x + b_j^i \qquad 2.4\text{-}13$$

where z, b, and w represent the output, bias weight, respectively, of the interconnected nodes in a deep learning-based artificial neural network. Even though deep learning can enhance the learning process compared to traditional machine learning, according to Zhang et al. (2019), it has several limitations that make it challenging to apply. First, it requires huge amounts of data to achieve its best performance, which makes it time-consuming and also consumes excessive amounts of computing resources during the learning stages. Second, many researchers consider it expensive to use because it requires high-performance hardware and huge memory storage space to run the model. Finally, it can be very difficult to use in real-life applications involving data that change over time (Zhang et al., 2019).

Rule-based learning is another machine learning technique for feature selection and estimation. The technique works on the principle of reducing the feature dimensionality, thus reducing the processing time for object identification or classification (Fatima et al., 2019). This is an approach that identifies, learns, or establishes rules to store, manipulate or apply. It applies some type of learning algorithm to instantly identify relevant rules instead of humans using previous domain information to establish rules and curate a rule set. This approach includes artificial immune systems, association rule learning, and learning classifier systems (Article et al., 2011). The most famous rule-based approach is the prism, which works on the principle of the target class. The target class consists of the attributes of the most important features that are required for classification. Rule-based learning involves a set of tasks and activities that can only be completed using a set of predefined rules or algorithms. The predefined rules may include procedural instructions, predefined algorithms, header files, and functions that assist researchers and programmers in performing their tasks.

Other machine learning tools belong to the class of optimised rule-based tools called learning classifier systems. These systems combine learning components (supervised, unsupervised, or semi-supervised labels) with discovery components (usually some sort of genetic algorithm) (Urbanowicz et al., 2017) and possess a detection module and a learning module. Learning classifier systems allow multifaceted resolution spaces to be divided into simpler portions. They use a group of context-dependent instructions that communally stock and use information in a piecewise way to make estimates (Urbanowicz & Moore, 2009). The learning classifiers use learning components to match the features in the input data set before feeding the output to rule-based genetic algorithms. This combination of learning components with genetic components makes learning classifier systems very efficient. For the first iteration, some predefined features are selected. Once the input images are fed into the system, the features are cross-correlated with the pre-entered features of the system using a match set routine. The matched features are considered optimal for the object classification before being fed to the rule-based block. A genetic algorithm is then applied for further refinement of the features, which are then fed iteratively into the next input for better object classification. Learning classifier systems are based on updating features that are also called fitness functions. The fitness functions are updated using equation 2.4-15:

$$F_j = F_j + \left(\left(\frac{P}{[A]}\right) - F_j\right) \qquad 2.4\text{-}14$$

where F represents the fitness function of iteration j, A represents the exploration iteration, and P represents the probability of correct classification.

Inductive-based learning is a form of machine-based learning that is logic oriented. This is a machine learning approach that utilises first-order logic to represent theories and information (Cropper et al., 2021). Since first-order logic is communicative and indicative, inductive logic programming precisely identifies tasks that include structured data and contextual information. Inductive-based learning can handle a wide range of machine learning tasks, including classification, regression, clustering, and reinforcement learning. In addition, it depends on logic for information illustration and cognitive purposes (Blockeel et al., 2011).

2.4.2 Machine Learning Models and algorithms

Various machine learning models and algorithms are described below.

2.4.2.1 Linear Regression

Linear regression belongs to the class of supervised machine learning class. Linear regression models predict target values using linear functions of independent variables. Linear regression is used in the approximation of real values, such as home prices and average sales (Ray, 2017). In conventional least-squares linear regression, the best-fitting linear relationship between the dependent variable and one or more independent variables is established by minimising the sum of the squares of the deviations of the independent variable observations from the straight line. Figure 2-11 illustrates the linear regression between a dependent variable and a single independent variable.

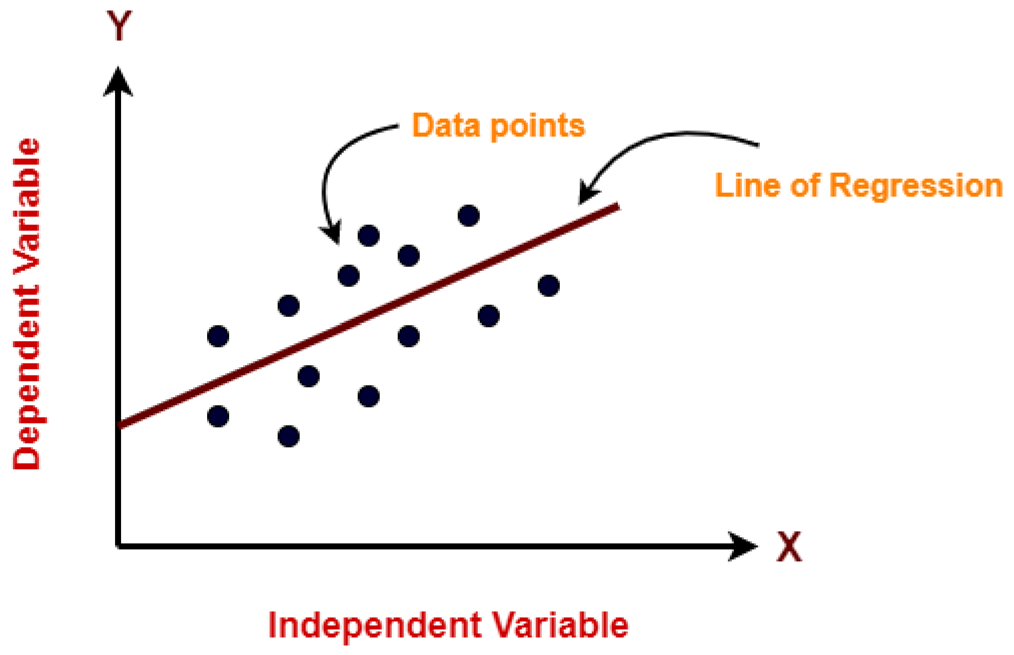


Figure 2-11: Linear regression concept adopted from (Kanade, 2022).

Linear regression is used to predict the value y (dependent variable) using the value x (independent variable). The identification of linear relationships between dependent and independent variables is known as linear regression. The main advantage of linear regression is forecasting and prediction of the unknown variables, thus assisting in classification of point of reference quickly. And it can be represented using equation 2.4-16.

$$Y = ax + b \qquad 2.4\text{-}15$$

where Y is the dependent variable, a is the slope of the regression line, and x is the independent variable, also known as the explanatory variable.

The main disadvantage of linear regression is that it does not describe non-linear relations well (Sindhu et al., 2021). Other disadvantages of linear regression are that it is very sensitive to outliers and that the data are assumed to be independent. It is, therefore, not well suited to modelling traffic phenomena using dependent (e.g., serial) observations (Sindhu et al., 2021).

2.4.2.2 **Decision Tree**

This is a supervised learning algorithm that is mainly utilized for categorizing problems. It is used in both categorical and constant dependent variables (Ray, 2017). Decision trees as the name imply work on the top-down phenomenon of trees-based machine learning algorithms that employ statistical modelling for object classification. The classification of data in this famous machine learning approach is obtained by arranging it in classes. The obtained results are represented through trees or flow charts. One class of data is obtained by classifying data in the dataset after traversing the set of queries from the top (root) to the bottom (leaf). The main role in object classification is served by the root, which is normally the attributes of the object, whereas the leaf usually represents the class.

The decision tree algorithm is applied by following several steps. The first step is to identify the attributes i.e., the training data to be encapsulated in the root. Then training data are divided based on selected features or attributes that depends on the application scenario. In this step, statistical measures are used to divide the attributes from the training data. Finally, partitioning is performed through recursive algorithms until all the training data are encapsulated in the root and divided into classes (Mohssen et al., 2016).

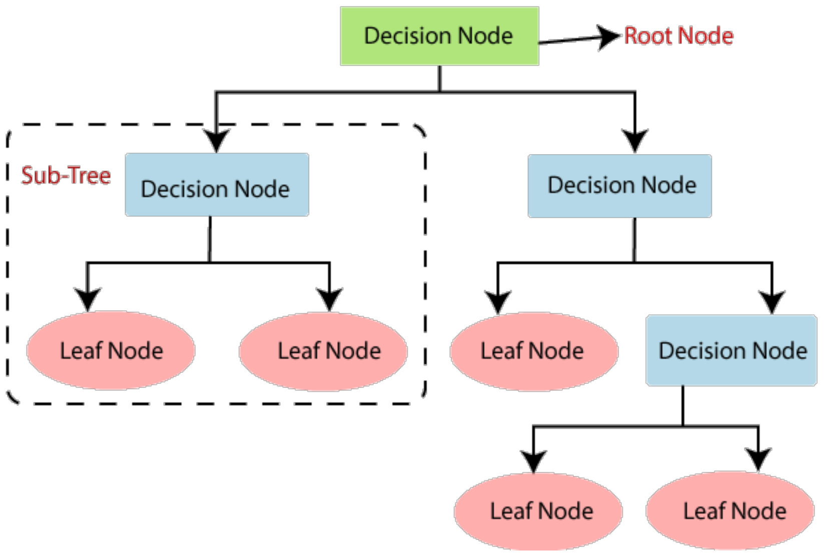


Figure 2-12: The concepts of the Decision Trees adopted from (Mohssen et al., 2016).

Figure 2-12 illustrates the decision tree concept. The decision lies with the root and the parent nodes at each level. The leaf nodes are classes that contain the data, while the root and the parent nodes classify the data based on the attributes of each object class. The decision trees are represented by Gain Ratio formula stated in equation 2.4-17:

$$Gain\ Ratio = \frac{Entropy\ before - \sum_{j=1}^{k} Entropy\ (j, after)}{\sum_{j=1}^{k} w_j log_2 w_j} \qquad 2.4\text{-}16$$

where entropy represents the uncertainty of a decision at specific tree level and w represents the weight of the node.

The decision trees have been used in many domains to perform anomaly detection and prediction tasks. However, the algorithm is restricted to the size of the dataset that can be trained as it does not perform well if the datasets is scaled or changed. Furthermore, if there are any missing data, the algorithm may not perform accurately. Finally, the algorithm requires high-performance hardware and substantial computer memory to produce accurate results (Çetinkaya & Horasan, 2021). These issues make decision trees a poor choice for traffic congestion detection and prediction, as the datasets are huge and change overtime.

2.4.2.3 **Support Vector Machine**

Support Vector Machine (SVM) can be used as both regression techniques and classification techniques, which explains why they are so popular. The amalgamation of two algorithms makes them more accurate and less resource intensive than their counterparts. In SVM classification, every data item is plotted as a point in dimensional space with the value of each characteristic being the value of a certain coordinate, as shown in Figure 2-13. SVM classifies objects into separate classes by seeking the best decision boundary between them. For instance, if there are two characteristics such as height and length of hair, these two variables are plotted in two-dimensional space, with every point representing two coordinates known as support vectors (Ray, 2017).

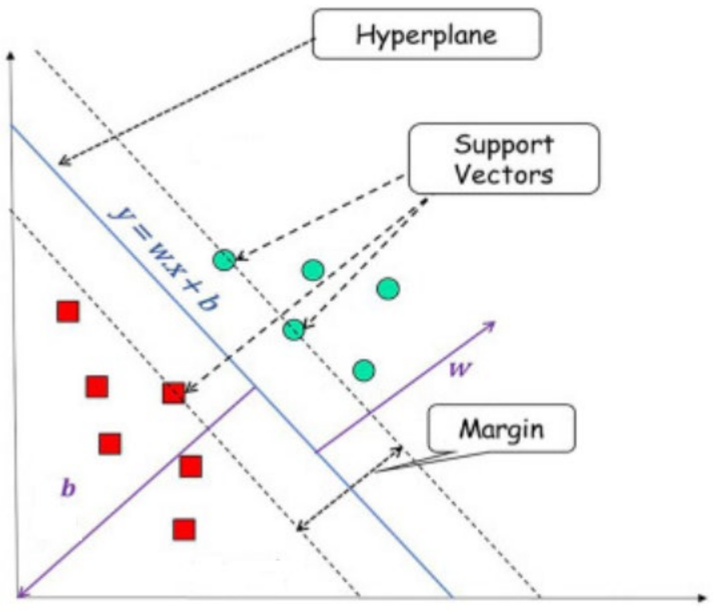


Figure 2-13: SVN working concepts adapted from (Rani et al., 2022).

The SVM can be represented by the equation 2.4-18 :

$$w^T x - b = 0 \qquad 2.4\text{-}17$$

where w is the normal vector to the hyperplane and b is the offset of the hyperplane.

Unlike linear regression or logistic regression techniques which find the optimal solutions through probabilities, SVM identifies the minimum distance among the data points and tries to keep the decision boundary large (Jun, 2021).

The main advantages of SVM are that it is fast and less computationally expensive than other algorithms described. It works very well when the data points are not clustered in a complex way . That is, it is efficient when data points belonging to different classes are clearly separated. SVM works best when the number of samples is less than the number of dimensions (Huang et al., 2018)

A limitation of SVM is that it is not suitable for data sets that are too large or are corrupted by noise. Noise makes the detection of data points and the creation of boundaries very difficult. SVM is also not suitable if the number of training samples is less than the number of features (Zixuan, 2019). Because SVM lacks the ability to adapt to new data streams and is not robust to noise, it was not implemented in this thesis research.

2.4.2.4 **K- Nearest Neighbour (KNN)**

The K-Nearest Neighbour, algorithm is a famous machine learning technique used for statistical analysis of data. This technique is used for classification and for solving regression problems. It is widely utilised for market-related classification problems.

KNN is a simple algorithm that has all available cases and classifies new cases through a major vote of its k neighbours. The case being assigned to the class is the most common amongst its k nearest neighbours, measured by a distance function (e.g., Minkowski, hamming distance, Manhattan).

KNN is a simple yet very powerful classification algorithm that works based on similarity. It is a non-parametric approach that classifies an object based on the degree of similarity between data points. The KNN formula is best represented by the Minkowski distance given by equation 2.4-19 below:

$$distance\ (x, z) = \left(\sum_{r=1}^{d} |x_r - z_r|^p\right)^{1/P} \qquad 2.4\text{-}18$$

where x and z are two neighbourhood identities, while p is the distance between them.

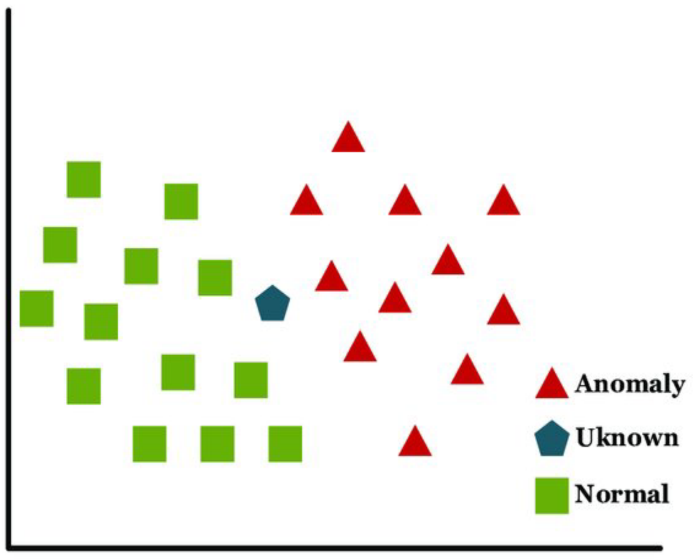


Figure 2-14: KNN concept in anomaly detection adopted from (Asharf et al., 2020).

The main steps of the KNN algorithm are as follows. First, the estimation parameter that determines the number of neighbours in the neighbourhood is identified. Next, the distance between the training samples and the query instances is calculated. The distances are then sorted in descending order and the nearest neighbour is determined based on the estimation parameter determined in the first step. The nearest neighbours are then categorised. Finally, the prediction value for classification is based on the majority of the nearest neighbours.

Figure 2-14 illustrates the concept of KNN in anomaly detection. The red data points represent anomalies, while the green data points represent normal data. When an unknown data element is considered, it is compared with its nearest neighbour to classify it based on the two categories: normal or anomaly (Asharf et al., 2020).

KNN algorithms can be used in almost any domain and have been applied to smart transport-related subdomains such as traffic detection and prediction in a number of studies. However, choosing K can be a challenge when conducting KNN modelling (Ray, 2017). KNN is one of the methods used for comparison in this research, as explained in Chapter 6.

2.4.2.5 **Random Forest**

The random forest algorithm, also known as the random decision forest algorithm, constructs a number of decision trees while training them to classify the objects into classes. It is a very well-known technique for classification and regression. For classification purposes, the output selected by the greatest number of trees is selected as the main class. For regression purposes, an individual tree's mean prediction or estimation is used. A random forest is an ensemble of decision trees. To classify a new item on the basis of its features, each tree provides a division, and it is assumed that the tree ‘votes’ for that class. The forest picks the division that has the most votes (Ray, 2017).

The main advantages of random forest techniques are that they are easy to understand, they have a rather simple structure, and they are more efficient than their counterparts (Savargiv et al., 2021).

Figure 2-15 shows the basic working principle of Random Forest and how it works as an ensemble classifier.

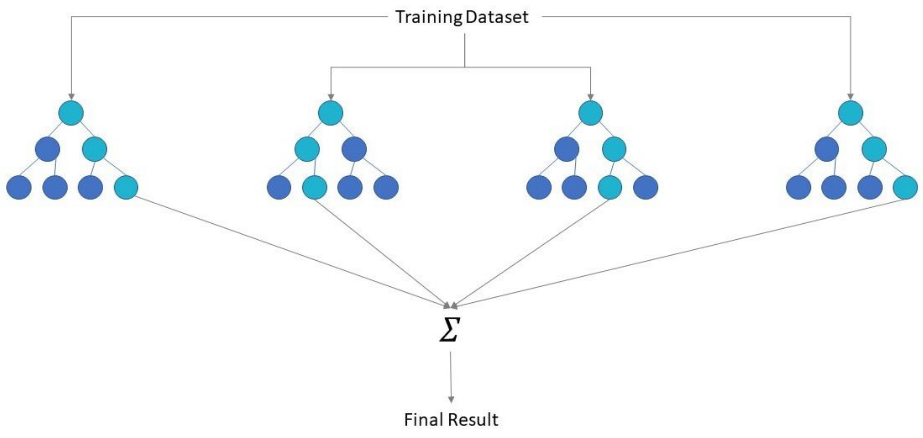


Figure 2-15: Random Forrest Classifier adopted from (Savargiv et al., 2021).

As clearly shown in the diagram, the random forest classifier uses bunch of decision trees for classification and regression. This is the reason why they are called forest instead of trees because of their ability to use multiple decision trees. The training data set is divided into multiple decision trees. The output of the decision trees is combined to yield the final result. Since the random Forrest classifiers use multiple decision trees, they can be represented by the same Gain Ratio formula as is used for decision trees (equation 2.4-20):

$$Gain\ Ratio = \frac{Entropy\ before - \sum_{j=1}^{k} Entropy\ (j, after)}{\sum_{j=1}^{k} w_j log_2 w_j} \qquad 2.4\text{-}19$$

where entropy represents the uncertainty of a decision at a specific tree level and w represents the weight of the node.

The main advantage of random forests is that they are less prone to overfitting, which is a major disadvantage of conventional decision trees because they are tightly bound by all the samples. The provision of multiple decision trees in random forests reduces the risk of overfitting. Another major advantage of random forests is flexibility since they perform both regression and classification in a single algorithm (Savargiv et al., 2021).

The disadvantages of random forests are that they are time-consuming and resource intensive. Since they require multiple decision trees makes the algorithm more complex in nature (Schonlau et al., 2020)

2.4.2.6 **Artificial Neural Networks (ANN)**

Artificial Neural Network (ANN), also commonly referred to simply as neural networks, are networks of nodes used for classification. ANNs simulate the manner in which the human brain assesses and processes knowledge. ANNs can be used to address problems that are challenging for human beings or statistical methods (Silva et.al., 2016). ANNs have self-learning capacities that allow them to generate outcomes more efficiently as more information becomes available. An ANN is designed like a human brain, consisting of neuron nodes which are interlinked in the form of a web and consists of thousands of artificial neurons known as processing units which are linked by nodes. The processing components consist of input and output units. Input units receive several types and structures of data on the basis of an internal weighting system. The purpose of the neural network is to identify the data to be presented in the output report (Frankenfiels, 2020).

The main applications of ANN are in the fields of pattern recognition and analysis, artificial intelligence, and digital image processing.

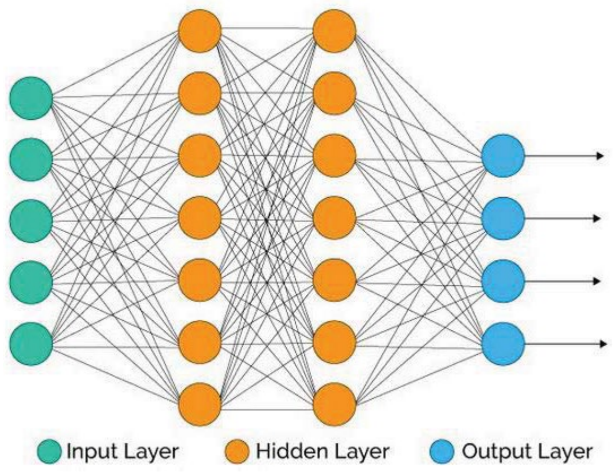


Figure 2-16: ANN Architecture adopted from (Dastres et al., 2021).

Figure 2-16 shows the basic architecture of an ANN, made up of several processing nodes that are interconnected in the form of a network. The nodes are called neurons. The neurons are arranged so that they remain connected. The input layer is responsible for receiving the input, and the output layer generates the output for classification. Each layer is assigned an appropriate weight while each neuron has a value with a predefined threshold. The input, output, and their relation to weight can be seen in Figure 2-16.

An ANN can be represented by the binary sigmodal function shown as equation 2.4-21:

$$f(x) = \frac{1}{1+e^{-x}} \qquad 2.4\text{-}20$$

The main advantage of ANNs is their adaptive learning capability: they possess the ability to adapt based on the input that they receive. For example, a trained neural network may fail to recognise a slight variation in handwriting patterns if it is trained on a specific handwriting style. A properly trained ANN model, however, will quickly adapt and will respond to any changes in the writing pattern (Dastres et al., 2021).

ANNs also have some disadvantages. One major disadvantage is the lack of proper guidelines for designing a uniform network. This trait makes initial training of the network impossible. Another disadvantage is that it is impossible to predict the uniformity of the network for future reference. In addition, an ANN needs considerable computer memory space to achieve its best performance, which could be challenging to implement in some real-life scenarios (Farizawani et al., 2020). In a traffic congestion application, for example, an ANN will be very complex to implement.

2.4.2.7 **Genetic Algorithms**

Genetic algorithms are models based on Charles Darwin's theory of natural selection. Genetic algorithms are used to solve complex problems and parallelism. In other words, they are used in solving optimisation problems in machine learning. GA models are important because they can solve difficult problems that would otherwise take a long time to solve. They are used in data centres, electronic circuit design, image processing, code breaking and artificial creativity (Mbaabu, 2021).

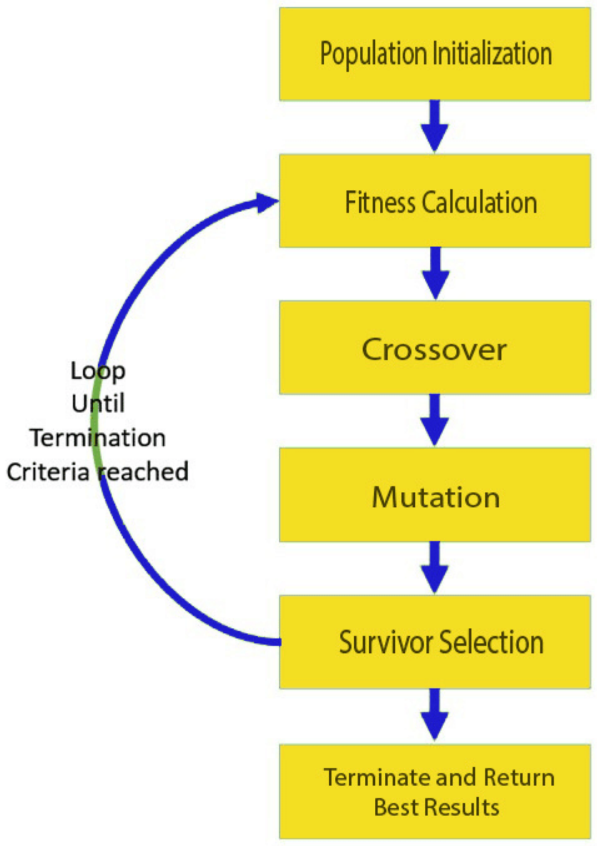

Population Initialization
Fitness Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best Results
Loop Until Termination Criteria reached

Figure 2-17: GA flowchart adopted from (Haldurai et al., 2016).

Figure 2-17 shows the basic working principle of genetic algorithms. Genetic algorithms iteratively produce a generation to identify the fittest individual possible (Sarmah, 2020). A problem or a set of problems is identified by an individual, and then a solution is proposed. A fitness function is then calculated against each rule to measure its adoption in a certain environment. As a genetic algorithm assumes the role of certain individuals, it can easily be seen as a set of individuals that are randomly generated (Haldurai et al., 2016). As the figure shows, the three basic steps in each iteration are mutation, crossover, and selection.

The main advantage of genetic algorithms is global optimisation and parallelism. Since genetic algorithms work on the basis of the selection of individuals rather than a sequential approach, parallelism can be achieved, which makes these algorithms faster. Another advantage is that they can provide multiple optimal solutions because of their probabilistic and random nature (Dipti et al., 2020). The main disadvantage of GAs is that since they deal with probabilities and randomness, they are more computationally intensive than other types of ML algorithms (Sarmah, 2020).

2.4.2.8 **Linear Classifier**

In machine learning, the main goal is to classify the object of interest based on certain parameters or characteristics. The linear classifiers perform this job of classification by categorising an object into a certain class based on the characteristics of parameters. Linear classifiers categorise data into labels on the basis of a linear amalgamation of input characteristics. Hence, linear classifiers divide information by means of a line, a plane or a hyperplane. However, these classifiers can only be used to categorize data that can be separated linearly. However, they can be modified to divide non-linearly dividable information (Alpaydin, 2020).

Linear classifiers can be divided into discriminative models and generative models. Generative models use probability density functions for object classification, whereas discriminative models employ conditional density functions. Examples of linear classifiers are naïve Bayes classifiers and linear discriminant analysis (LDA).

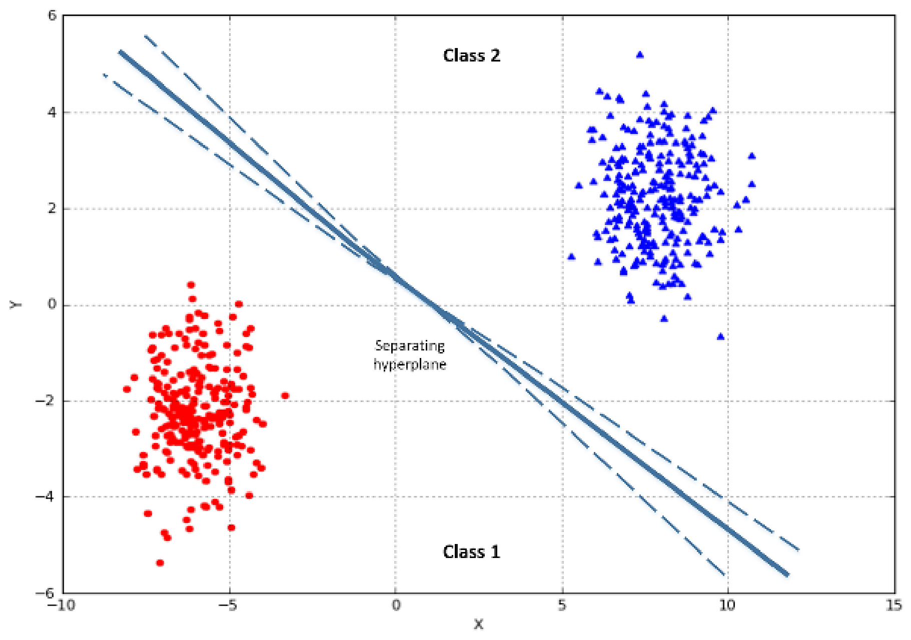


Figure 2-18: Linear Classifiers working phenomenon (Alpaydin, 2020).

Figure 2-18 shows an example of a linear classifier. As the figure shows, linear classifiers classify data points based on a set of characteristics. Linear classifiers can be represented by the simple thresholding function shown as equation 2.4-22.

$$f(x) = \begin{Bmatrix} 1 & if\ w^T x > \emptyset \\ 0 & otherwise \end{Bmatrix} \qquad 2.4\text{-}21$$

The main advantage of linear classifiers is that they are less computationally intensive than other algorithms and less complex in nature. The main disadvantage of linear classifiers is that if the input data are non-Gaussian in nature, the linear classifier tends to fail (van Engelen & Hoos, 2020).

2.4.2.9 **Hierarchical Clustering**

Hierarchical clustering is a type of algorithm that divides items that are alike into clusters or groups. The result is a group of clusters in which every cluster is different from the others. In addition, the items in every cluster are largely identical to one another. Hierarchical data is used with either a distance matrix or raw data. When raw data are provided, the software will compute a distance matrix in the background.

Hierarchical clustering works in a way that it begins by treating each observation as a different cluster. The following steps are also frequently executed:

- Recognising the two clusters that are closest together.
- Combiningthe two most identical clusters.

These processes progress until all of the clusters are joined together (Awad & Khanna, 2015b).

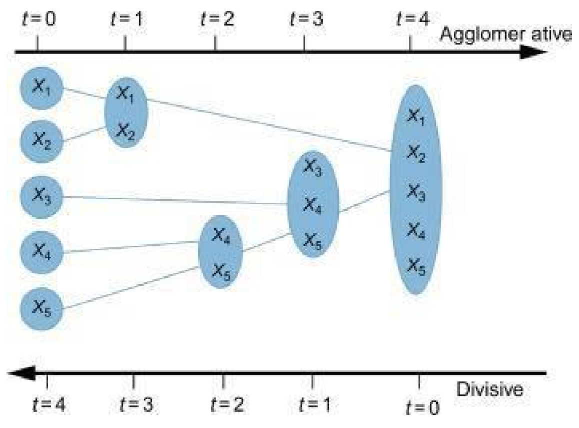


Figure 2-19: Hierarchical Clustering concept adopted from (Shetty & Singh, 2021).

Figure 2-19 shows the hierarchical clustering process. Data points are clustered based on common attributes. At each point, new clusters are made until an object is classified based on a single cluster. Hierarchical clustering can be represented by equation 2.4-23 below:

$$D(r,s) = \frac{T_{rs}}{(N_r * N_s)} \qquad 2.4\text{-}22$$

where T_{rs} is the sum of all pairwise distances between cluster r and cluster s. N_r and N_s are the sizes of the clusters r and s, respectively.

The advantages of hierarchical clustering are that it is easy to implement and less complex than several other algorithms. Its main disadvantage is that it cannot handle huge amounts of data without tending to fail (Shetty & Singh, 2021).

2.4.2.10 **Cortical Algorithms**

Cortical algorithms are a class of machine learning algorithms that are inspired by the human visual cortex. The algorithm stores a pattern of data in a rather invariant form and then recalls it whenever required. It supplies arrangements of patterns in an invariant system which memorises those patternsauto associatively. CA is designed in such a way that it has mini columns of different sizes, resembling the structure of the human brain. In this case, a mini column refers to a group of neurons that share a similar receptive field (Awad & Khanna, 2015a).

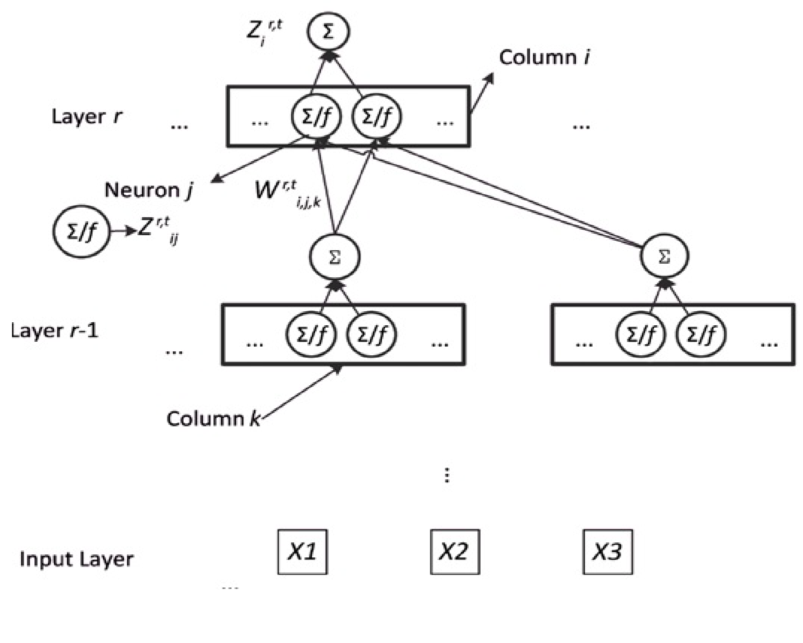


Figure 2-20: Cortical Algorithms adopted from (Awad & Khanna, 2015a).

Figure 2-20 is a schematic diagram of a cortical algorithm. The connections in the diagram do not represent the data being carried by the connections; rather, they represent the connections between classes. These connections function similarly to neurons for carrying data. Cortical algorithms are two-step algorithms. In the first step, features are identified, and in the second stage, connections between features are identified (Awad & Khanna, 2015a). Cortical algorithms can be represented using equation 2.4-24 below.

$$Z_i^{r,t} = \sum_{j=1}^{M} Z_{i,j}^{r,t} \qquad 2.4\text{-}23$$

The main advantage of cortical algorithms is that they are very precise in nature, while the main disadvantage is that they are very precise; the main disadvantages are that they are very complex and non-optimal. Their complexity makes them difficult to adopt. Cortical algorithms are adopted as the computational implementation of HTM, as discussed in more detail in Chapter 3.

2.4.2.11 **Learning Algorithm**

Learning algorithms are also called imitating algorithms since they possess human-like abilities to update themselves over time without human intervention when provided with additional data. These algorithms are used to divide cases on the basis of the similarities of their feature, patterns, naturally appearing trends, or information relations. This category of algorithms includes clusturing and self-organising maps (Devijver, 2020).

The main advantage of learning algorithms is that they are very precise because they are programmed to mimic human activity. The other advantage is that they can be optimised easily by inputting more data. Their main disadvantage is that they are very generic in nature; a conclusive and particular solution is very difficult to attain.

2.4.2.12 **Anomaly Detection**

Anomaly detection is a type of machine learning algorithm based on identifying an anomaly while also working on finding an optimised solution. The main purpose of such algorithms is to identify corner cases and rare occurrences in the service of arriving at an optimised solution. They are normally used together with other optimisation algorithms.

Anomaly detection is also a type of data mining technique that identifies data points, events, and observations that deviate from a dataset's normal behaviour. Anomalous data can reveal critical incidents, such as technical challenges to probable opportunities, such as variation in customer choices (Cohen, 2021).

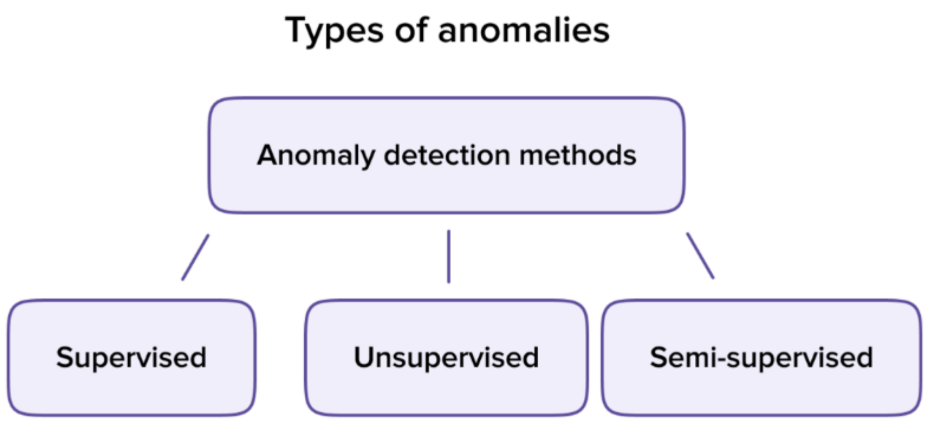


Figure 2-21: Types of Anomaly Detections adopted from (Gavrilova, 2021).

Figure 2-21 shows that anomaly detection algorithms can be categorised into any of the three conventional algorithm classes described in section 2.4.1, i.e., supervised, semi-supervised, and unsupervised.

The main advantage of anomaly detection algorithms is that they are very easy to use in conjunction with other algorithms, which makes them very easily adaptable. Another advantage is that they enable researchers to focus on the problem at hand rather than on finding errors (Das Nath & Bhattasali, 2020). The work described in this thesis is an implementation of an anomaly detection framework to identify traffic congestion when it occurs.

2.4.2.13 **Hierarchical temporal memory**

Hierarchical temporal memory (HTM) is a technique inspired by the structure of the human neocortex that attempts to mimic how the human brain processes information acquired through vision, hearing, and other means, leading to memory formation and prediction of events and conditions.The HTM system is more trained than programmed and the basic learning algorithms used in it are not specified in certain sensory domains and may be used for a wide set of problems involving modelling complex sensory data (Chen et al., 2012). HTM has been on a long journey from its emergence to the present day. This progress is due to the persistence of Jeff Hawkins, a neuroscience investigator pursuing his dream of building smart machines. He believes that the only way to build smart systems is to look at how the brain works, especially its structure. Compared to the other algorithms discussed in this section, HTM seems more reliable in terms of memory usage and handling huge streams of data. Therefore, the framework described in this thesis was implemented using an HTM algorithm, as explained in Chapter 3.

2.4.3 Machine Learning Models Assessment Measures

Machine learning models should provide precise predictions to provide real value. How the model generalises based on unseen information is a significant feature of all machine learning models (Mutuvi, 2019). It is important to determine whether the model actually works and whether its predictions are reliable. Certain methods can be utilised to assess the effectiveness of a machine learning model and determine whether it generalises to new, previously unseen information.

The purpose of model assessment is to measure the preciseness of a model in generalising future information. The techniques for assessing the performance of a model are categorised as holdout or cross-validation techniques.

Table 2-1 summarises the main differences between holdout and cross-validation techniques. Holdout techniques tend to split data sets into testing and training subsets, while cross-validation models tend to organise the data into k groups. Cross-validation is typically the preferred choice because it provides more flexibility by organising the data into groups. Both types of techniques are tests, which means that the information is not seen by the model (Mutuvi, 2019). Cross-validation involves dividing the initial observation dataset into a training set used to train the model and an independent test set used for assessment (Awwalu, 2019). The most prominent cross-validation method is k-fold cross-validation in which the initial dataset is divided into k subsets (Mutuvi, 2019).

Table 2-1: Holdout versus Cross Validation.

	Hold Out	**Cross Validation**
Data arrangement	Testing and training frames	Data organized in k-groups

	Hold Out	**Cross Validation**
Accuracy	Less accurate	More accurate
Flexibility	Less flexible	More flexible
Ease of Use	Less complex	More complex
Time consuming	Less time consuming	More time consuming

The aim of holdout assessment is to test a model using information other than that on which it was trained. This produces an unbiased approximation of learning performance. The dataset is arbitrarily divided into the following three subsets:

- **Training set:** This is a subset of the dataset that is used to create projecting models.
- **Validation set:** This is a subset of the dataset that is used to evaluate the outcome of the model created in the training stage. It offers a platform for a test for finetuning a model's parameters and selecting the most significant performing model. Nevertheless, not all modelling algorithms require a validation set.
- **Test set for unseen data:** This is a subset of the dataset that is used to evaluate the probable future outcome of a model. If a model fits the training set better than the test set, overfitting is most likely the cause.

The holdout technique is important because its fast, simple, and flexible. Nevertheless, this method is sometimes linked with increased variability because variations in the training and test dataset can lead to important differences in accuracy.

2.5 ML in the Context of Smart Transport

2.5.1 Types or Approaches

ML has the ability to provide live traffic prediction in real time, future traffic prediction and short-term traffic prediction based on current observations and historical information (Deekshetha et al., 2022). Hence, ML is an appropriate approach to deriving hidden insights by means of iteratively learning from data without being explicitly programmed. This can be accomplished using approaches such as GPS and the Gaussian method (Kamble & Kounte, 2020).

2.5.2 Role of ML in Anomaly Detection

The use of machine learning in anomaly detection requires a proper comprehension of the issue, especially in conditions involving unstructured data. When data are structured, it means that there is an understanding of the issue. Anomalous data may be easy to identify because pre-existing rules are violated in structured data such as fraud detection applications (Johnson, 2020).

On the other hand, unstructured and obscure information such as images encoded as a sequence of pixels have subtle interpretations that may make conventional algorithms irrelevant. In any case, the structure can be found in the last layers of a convolutional neural network (CNN) or in any figure using algorithms.

As explained in section 2.4.2.12, anomaly detection can be supervised or unsupervised, depending on the nature of the information.

2.5.2.1 **Supervised**

In supervised anomaly detection, training data are labelled as normal or anomalous. In a supervised setting, a dataset is neatly prepared for a data scientist, and all of the information is labelled as nominal or anomalous. All anomalous data are known ahead of time (Johnson 2020). This means that there are sets of information that are anomalous but are not distinguished as such so that the model can be trained.

2.5.2.2 **Unsupervised**

In unsupervised anomaly detection, the training information is not labelled and contains both nominal and anomalous points. The model is required to reveal to the modeler the difference between anomalous and nominal data. In a supervised scenario, a distinct set of items is required to establish order in the unstructured information (Johnson, 2020). Thus, in unstructured information, the main goal is to establish clusters out of the information followed by establishing few groups that information does not belong to. Some of the popular machine learning algorithms for unstructured information include self-organizing maps (SOM), k-means clustering, and one-class support vector machine. SOM is a type of machine learning algorithm which uses fewer dimensions to represent multi-dimensional data and ensures that the topological structure of the data remains intact. The main advantage of SOM is very evident: it can reduce the complexity of the algorithms to a level at which they can be interpreted easily. The drawback, however, is that they need plenty of neurons for data representation in clusters (Lakshminarayanan et al., 2020).

Another example is k-means clustering, which belongs to the class of unsupervised machine learning algorithms and is considered to be one of the most powerful algorithms. K-means clustering categorises data points in prespecified clusters even when very little relevant information is available about them. First, the cluster centres are identified. This is done by taking the arithmetic mean of the available data points. Next, the incoming data points are attached to one of the available clusters based on a specific set of features. The k-means clustering algorithm is fast, simple, and robust. The main disadvantage is that since the output is dependent on the number of input clusters, sometimes it becomes very difficult to predict the original number of clusters (Ahmed et al., 2020).

Another example of unsupervised learning methods is the one-class support vector machine, which finds novelty in a system i.e., it traverses the pool of new data points, and based on a specific set of features, adds them to the pool of existing clusters. Similar to SVM, one-class SVM works best when there is a clear boundary between data points. The main disadvantage is that it fail to deal effectively with large data sets (Seliya et al., 2021).

2.6 Role of ML in Management of Traffic Congestion

Traffic jams cause congestion, increase trip times, cause delays in the performance of daily routines, increase pollution, and increase road user costs. Many cities are seeking to develop sustainable transport systems and the logistics to solve traffic issues through technology (Badar 2020). Machine learning and deep learning techniques offer promise for solving traffic congestion problems by contributing to the establishment of intelligent and adaptive traffic control systems.

There are various trends evident in the use of machine learning algorithms in the context of smart transportation (Badar, 2020)

As discussed in section 2.4.2.4, KNN is widely used for classification and for solving regression problems. Instance-based KNN is a variant of the conventional KNN technique except that the algorithm assigns data points to the nearest neighbour dynamically. KNN is used for multi-instance-based detection of the computational time (Lin et al., 2022). KNN is a classifier specifically used in supervised learning. However, in the case of instance-based training, where many hyper-parameters are used, conventional KNN is combined with the graph neural network technique. This combination is the best for dealing with instance-based algorithms and can be applied in real-time independent accident detection (Lin et al., 2022). The limitation of this algorithm is that it usually used with dimensions algorithms such as ANN, and thus the algorithm comes with too many dimensions, which makes it very complex in nature (Lin et al., 2022). The concept of SVM is explained in general in section 2.4.2.3. Here, SVM is explained in the context of identifying road traffic abnormalities. In a study conducted by Chen et al. (2022) to implement a parking space detection system, information from cameras installed at parking lots was accessed, and images were pre-processed to extract the most relevant features. These images were transformed into greyscale images for detection using SVM was combined with a multi-layer perceptron neural network (MLP-NN). According to Chen et al. (2022), the results were 93% accurate. However, these results could be unreliable, as the authors stated that it is difficult to check highway roads on a daily basis, and an alternative way to address this issue is needed (Chen et al., 2022). SVM extracts the parameters for use in the algorithm from the traffic stream and then tries to reduce the dimensions of the data (Huang, 2022). Unlike other optimisation methods, SVM adapts to data from a lane-level traffic stream and uses it for classification purposes (Huang, 2022). The limitation of this approach, according to Huang (2022), is that if the data set is too large, meaning that traffic at the lane level exhibits severe congestion, the system may fail.

Regression trees are very effective at prediction and estimation. Traffic data are collected in a city over a specific period, and based on the data, a regression tree algorithm is applied to predict the number of traffic mishaps (Tamir et al., 2021). The limitation of this approach is that the accuracy of the system may be limited if previously obtained data are not analysed carefully. This approach may be used to forecast short-term and long-term traffic flow in work zones.

2.7 Intelligent/Smart Transport

2.7.1 Definition and History

Urban centres witnessed rapid growth in the early 20th century with significant migrations and increase in real estate. development. Many urban problems have emerged as a result of the growth of urban areas. Traffic congestion is a phenomenon associated with transportation that leads to slower speeds, longer trip times, and increased vehicular queuing. Urban road networks have experienced increases in traffic volumes since the 1950s. A number of definitions have been proposed for traffic congestion. The basic definition of congestion is traffic flow that occurs when demand exceeds road capacity (Arnott et al., 2005). Congestion is also defined from a delay–travel time perspective as occurring when normal traffic is disrupted by high densities of vehicles that increase travel time (Downs, 2015). Another definition of congestion is increased road user costs as a result of disruption in traffic flows (Li et al, 2020). The common theme of these definitions is that congestion occurs for multiple reasons, including excessive numbers of vehicles during peak hours, weather conditions, work zones, special events, and accidents. Rapid urbanisation and increases in real incomes have led to increased vehicle ownership (Li et al, 2020). Furthermore, the existing roads in some urban areas cannot accommodate the growth in traffic volumes that accompanies population growth, resulting in road capacity being exceeded during peak hours. Traffic incidents, road work, and weather events can also temporarily disrupt traffic and cause congestion. Urban policy makers, government agencies, and other stakeholders have proposed numerous measures to reduce traffic congestion. They have also sought to develop various models for measuring and quantifying traffic congestion (Nagy et al, 2021).

2.7.2 Causes of Congestion and its Negative Aspects

Traffic congestion has become a perennial aspect of urban life and has multiple causes. Agyapong & Ojo (2018) found that traffic congestion can occur as a consequence of traffic disturbances that are temporary occurrences and only have momentary impact on traffic. Incidents such as accidents, harsh weather conditions, and road construction are the most common reasons for congestion. Kozlak & Wach (2018) observed that that traffic disturbances such as accidents can cause road blockages and disrupt traffic flow because they create confusion among drivers. Poor weather conditions can also result in vehicles slowing their pace because of safety concerns, which can also result in traffic congestion.

Onyenke et al. (2018) observed that network overload occurs due to reduced capacity of roads or increased demand for transportation. Bottlenecks can cause congestion. Das & Nayyar (2019) postulates that bottlenecks, like narrowed roads and clogged road intersections contribute to congestion. Specifically, lanes merging with existing roads and reductions in the number of lanes can cause congestion. Falcocchio & Levinson (2015) concluded that traffic congestion occurs because of demand overwhelming the capacity of existing roads. Population growth and increases in the number of vehicles in use are demand factors that increase the demand for roads.

The negative aspects of traffic congestion have been studied extensively. Kong et al. (2016) cites wasted time for motorists and passengers, delays, and inability to forecast travel time accurately as negative implications of congestion. Suresh et al. (2018) highlights wasted fuel causing pollution and carbon dioxide emissions as negative aspects. Furthermore, Wen & Lai (2017) cites high levels of stress and frustration among motorists, wear and tear on vehicles, interference with emergency services, and spill over effect as motorists use secondary roads and side streets as alternative routes.

Identifying the causes of congestion can help to improve the quality of traffic management systems and aid researchers in developing ways to reduce traffic congestion that take all its causes into count. The negative effects of congestion have been studied not only in the transport domain but in other domains such as heath and economics (Samal et al., 2021; Moyano et al., 2021).

2.7.3 Congestion Prediction

Traffic congestion in urban areas causes losses and obstacles to the economy, commuters, and environment. Forecasting the congestion levels of road networks in a timely manner can prevent its formation and increase efficiency and capacity of road networks (Elfar et al, 2018). The aim of congestion prediction is to forecast real-time traffic information using floating car data and historical traffic data. Congestion prediction is dependent on data because updated and accurate traffic information needs to be available (Akhtar et al 2021).

Therefore, one of the objectives of this thesis is to study available machine learning models that have been applied to smart transport systems to improve their performance. In Section 2.4.2, the concepts, advantages, and limitation of various machine learning models were discussed. In this section, studies related to the work done for this thesis are reviewed.

Kumar et al (2018) reviewed a number of technologies that can provide real-time and adaptive capabilities for traffic department in managing congestions. This study is important because it highlights the use of insights derived from real-time traffic management in predicting congestion levels in urban areas. Moreover, the study recommends using edge data analytics, which is an advanced machine learning approach to analysis of data produced and consumed by edge devices as one of the ways to improve traffic and monitoring congestion levels. However, using edge computing requires a very powerful machine to deal with the huge amount of data involved without excessive use of computational resources. The study concluded that connected and dynamic city environments need robust and innovative congestion prediction models are not resource-intensive and are not expensive to use (Kumar et al, 2018). Therefore, one of the research questions highlighted in Section 1.5 and addressed in this thesis is to whether HTM can be successfully applied, as it requires less memory than other models and can deal with huge amounts of data, as explained in Chapter 3.

Ranjan et al (2020) propose an efficient traffic congestion model that utilizes data from open source and online web services. The study also uses a hybrid neural architecture that helps to obtain spatial (images) and temporal (video sequences in images) information to predict system congestion levels. Spatial data can be obtained from an aerial view or any direction. Smart traffic systems are used to obtain image-related data. In the case of a traffic management system, temporal (time-related) data can be item such as the times at which images are obtained, vehicle speeds, and the quantity of images available at a given time. The use of temporal–spatial data is recommended for efficient automated traffic management (Zhu et al., 2022) because it enhances computational efficiency and prediction performance. The limitation of the approaches presented by Ranjan et al (2020) is the lack of adaptability to changing congestion conditions over time.

Another study by Chen et al (2018) developed a model for predicting traffic congestion that utilizes deep convolutional neural networks. The model predicts periodic traffic data for short-term congestions. Local temporal characteristics and multiscale traffic patterns are used to address global scale of congestion (Chen et al, 2018). The advantage of the model is that it helps in addressing short-term and long-term traffic congestion. However, if new data attributes are used then more time and resources are consumed.

Chiabaut & Faitout (2021) worked on real time estimation in congestion control using the clustering technique. Data from a 10-km section of the M6 highway near Lyon, France, were used. Chiabaut & Faitout (2021) split the dataset, using 25% for testing and 75% for training. A Gaussian mixture model (GMM), which is a mathematical approach to multiple clustering, was then applied (Chiabaut & Faitout, 2021). The KNN approach was also implemented because it is well suited to clustering congestion-based images. The models were not compared, and their accuracy was not reported (Chiabaut & Faitout, 2021).

Akhtar & Moridpour (2021) presented multiple AI models based on traffic patterns for use in reducing traffic congestion and thereby controlling the outcomes of accidents in the future. Once traffic patterns were analysed, the data points were assigned to the nearest neighbourhood through Fuzzy c-means clustering. The limitation of this technique is that since it is fuzzy-oriented, the clusters cannot be determined at the onset, which sometimes increases the complexity and execution time.

Sun et al (2019) developed a model for predicting traffic congestion that uses process GPS trajectory data. The trajectory data were matched with the road network. Moreover, the average speed of road sections were evaluated by the model (Sun et al, 2019). The model utilizes deep learning to achieve superior accuracy rates with respect to traffic congestion. The limitation of the proposed algorithm by Sun et al (2019) is that it increases computational complexity because of the inclusion of deep learning analysis and the time needed to assess the performance of the model.

Di et al (2019) developed a model that calculates the account spatial correlation between road segments to predict traffic congestion. It helps to predict congestion levels at various road segments. Moreover, it uses a spatial matrix that uses congestion propagation and spatial correlation found in road segments (Di et al, 2019). The disadvantage of the technique is the factors affecting spatial correlation. Spatial correlation involves the correlation of images, therefore low-resolution images and the presence of noise may affect the overall outcome of the algorithm.

Yang et al (2019) conducted research about the use of data science technologies in predicting traffic congestion. The findings propose an algorithm that uses occupancy rate of roads at a specific time as a variable to identify and evaluate congestion levels. This algorithm is effective in rapid and accurate evaluation of congestion. The limitation of the algorithm is the lack of fuzzy logic analysis routines in it. The algorithm lacks the ability to optimize itself in case of congestion due to irregular events happens i.e., accidents, etc.

Kumar & Raubal (2021) presented deep neural network techniques for congestion detection and estimation, as well as a method for using fully connected layered neural network structures to estimate traffic congestion using high-resolution images. Although the method shows very promising results, the use of layered structures makes the system more cost intensive.

Neelakandan et al. (2021) developed an intelligent traffic management system for predicting congestion on roads. According to their results, conventional machine learning algorithms such as toot mean square entropy (RMSE), Elman neural networks (ENN), and ANNs are not sufficiently capable of classifying congested road segments. Therefore, Neelakandan et al. (2021) developed an Internet of Things-based smart city system, which they named an Intelligent Traffic Management System (ITMS). This system detects locations of congestion and optimises traffic controls to optimise traffic flow. For that purpose, Neelakandan et al. (2021) used road traffic data from the Kaggle dataset and extracted data on three types of features: traffic data, place data, and weather data were extracted. They also proposed an optimized weighted Elman neural network (OWENN) algorithm, which is an advanced form of an ENN algorithm with more than one hidden processing layer with many neurons. The proposed model was used to classify images in the datast and identify congested and non-congested road segments. For non-congested segments, the authors proposed an IoT-based algorithm called an improved beetle swarm optimization (IBSO) algorithm (Neelakandan et al., 2021). This instance-based algorithm optimises traffic management using a dynamic programming approach to avoid repetitive calculation of instance-based phases. Finally a comparison of OWENN with ENN, ANN, RMSE, and MSE was conducted, and the results show that the model was up to 98% acccurate. The model operates automaticcally with the help of IoT technology, but the energy and data usage mechanisms are very costly (Neelakandan et al., 2021).

It can be concluded that traffic congestion prediction is an emerging field with various models being used to manage and mitigate the impact of congestion on roads and highways in urban areas. Each model seeks to develop optimised strategies that can help to ensure smooth traffic flow and reduce losses to the economy, environment, and commuters. However, the studies reviewed in this section mostly rely on complex models that can be expensive and time-consuming to use, which may affectthe ability of these models to predict congestion accurately.

2.7.4 Incident Detection

Incident detection is performed using technological, organisational, and human resources. Furthermore, it relies on inputs such as sensing and camera data, radio reports, cell phone calls, and others, as explained in section 2.7.2. In this section, related work in incident detection is discussed.

Lin et al (2020) noted that traffic data is collected with sensors that use microwave, laser, inductive and other loop detectors. Incident detection uses traffic characteristics such as average speed, traffic volume, and vehicle occupancy. Aggregated measurements from a single time interval known as datapoint are used (Lin et al, 2020). A collection of datapoints is known as a dataset.

Liu et al. (2019) found that incident detection is dependent on road management conditions, geometric design, environmental conditions, distance between sensors, time, and severity. Specifically, an effective incident detection system must have norms with time-dependent functions that integrate normal traffic patterns for given times (Liu et al, 2019). Advanced wavelet analysis and pattern recognition techniques have also been used in incident detection. Research shows that wavelet analysis helps to categorise and augment raw traffic data that is classified using a radial-basis-function neural network (Garg et al, 2017). Consequently, traffic patterns can be represented in ways that help to define incident and non-incident traffic conditions (Garg et al, 2017). A long time-series pattern can be normalised to eliminate false alarms generated by the system. Detection and false alarm elimination can be done efficiently using the system described.

One of the first analyses of a modular architecture that could be applied to freeway and arterial road algorithms was by Zhang & Taylor (2006). Bayesian rules were designed to store general expert traffic knowledge and be able to detect incidents universally. The freeway algorithm was evaluated using a virtual environment with traffic-related components implement to assess actual results. The simulation data were used to test and evaluate the freeway algorithm and the arterial road algorithm proposed by Zhang & Taylor (2006). The above incident detection framework is important because the detection rate and false alarm rate are insensitive to incident decision-making and because its development encouraged other researchers to proposed improved frameworks for incident detection. Research has also been done a catastrophe theory-based method for incident detection that analyzes sudden discrete changes in one variable while other variables function in smooth way (Saifuzzaman et al, 2018). Alarms are generated when speed drops considerably without a matching increase in occupancy and flow. The implications are that incident detection systems can distinguish between incidents and recurring congestion. The benefit of the catastrophe method is the use of multiple variables that can be compared with previous data trends to detect congestion on roads (Saifuzzaman et al, 2018).

Khan et al.(2021) investigated how the proficiency and quality of continuous incident identification models influence drivers' well-being and functional circumstances. The development of cloud-based quantum registering frameworks and developments in boisterous halfway-scale quantum gadgets have initiated another period of quantum-upgraded calculations that can be utilised to develop constant incident identification precision further. In this exploration, a hybrid AI model which incorporates traditional and quantum AI (ML) models was created to recognise incidents utilising the associated vehicle (CV) information. The incident discovery execution of the hybrid model was considered in contrast to pattern old style ML models. The system was assessed utilising information from a microsimulation instrument for various incident situations. The results demonstrate that a hybrid neural network containing a four-qubit quantum layer outperforms any remaining gauge models when there is an absence of preparing information. Furthermore, three datasets have been created by Khan et al.(2021) one with sufficient preparation, while the others prepared in insufficient way. The model achieved more than 98.9% for the first dataset. The crossover model improved the F2-score by 1.9% when compared to the traditional models. This demonstrates that even with limited information, as is common in CVs, the hybrid ML model outperforms conventional models. With improvements to the quantum registering framework, quantum ML models could be a promising option for CV-related applications when available data is insufficient.Research finds that statistical methods can be utilized to generate real-time data for comparison with predicted data (Islam, 2019). Incident detection can be done by identifying any variation in traffic flow values. A number of methods, such as stochastic methods, SND (Standard Normal Deviation) algorithms, DES (Double Exponential Smoothing) methods, and others are used in the above approach. The review of the literature has revealed numerous incident detection methods that have been proposed by researchers and applied by policy makers. There is a continuous drive to improve incident detection methods and help to manage traffic congestion in urban areas.

Xia et al. (2022) worked on traffic system improvement by implementing auto-detection of incident in real time. For that purpose, Xia et al. (2022) used YOLO (Python library to process live pictures from CCTV camera), along with other basic python libraries. They have used Ubuntu Operating system with 8 GB of RAM. The implemented dataset is KITTI dataset, which lack many aspects so they have pre-processed the dataset and arrange the pictures into organized form. The implemented model is convolutional neural network. The result shows that the model is implemented for the detection of live images. However, the research is lacking the evaluation matrix to understand every important aspect of the model.

2.7.5 Traffic Modelling

Modelling has become an instrumental aspect of traffic engineering in recent decades. Microscopic and macroscopic traffic models are commonly used in simulation tools to help traffic managers to simulate traffic states. Furthermore, traffic models can be used to generate simulated traffic data for use in analysis. However, one of the contributions of this thesis is a critical review of actual (not simulated) datasets available, as explained in Chapter 4, as actual datasets rather than simulation tools were used in this thesis research to test the proposed framework. Therefore, traffic modelling tools and methods are outside the scope of this thesis.

Macroscopic traffic modelling describes the overall state of a system for purposes such as simulating large numbers of vehicles on highways and urban areas. Macroscopic modelling represents the overall positioning of objects irrespective of the variances involved, which makes it more complex than microscopic modelling. Microscopic traffic modelling is better suited for low-volume one-lane or two-lane roads managed by small administrative services. Macroscopic traffic models are observed by regional authorities, whereas microscopic models describe the state of individual agents. Additionally, the simulation of urban-area traffic normally does not use equations but rather relies on agent-based models represented by finite-state machines that represent the longitudinal and lateral behaviour of objects (Zhang et al., 2021). Microscopic models work on the basis of interaction between individual vehicles, while macroscopic models consider the cumulative traffic flow within a particular area. Since macroscopic models deal with a particular locality, data from macroscopic models may be fed to microscopic models for the analysis of individual road segments (Erdelyi et al., 2022). Macroscopic traffic flow models describe relationships among traffic flow characteristics, such as density, flow, and mean speed (Gora & Rub, 2016). Microscopic traffic flow models can identify traffic patterns by evaluating the behaviour of individual vehicles in terms of properties such as their positions and velocities (Gora & Rub, 2016).

Peter (2012) found that traffic flow modelling relies on theoretical foundations such as network theory and kinematic wave modelling. In other words, the throughput of mobile units per time and transportation medium capacity is expressed by traffic models. The researcher found that effective models help researchers and engineers to find ways to optimise traffic flow and reduce the number of traffic jams.

Mehdizadeh et al (2020) found predictive or explanatory models to be efficient in achieving safe routing and facilitating the detection of traffic incidents. These models can be used to understand and quantify traffic incidents based on various driving conditions.

Haddouch et al (2018) developed a traffic model that combines a Petri net model with dynamic estimation of intersection turning movements. In other words, turning movement counts can be an important variable in predicting traffic flows.

Golovnin et al (2019) developed a microscopic traffic flow simulation model that uses web browsers for analysis and evaluation. The system is effective in detecting incidents and ensuring that traffic continues to flow efficiently. Oberoi et al (2017) proposed a qualitative model that assists in understanding spatial evolution of urban road traffic. In other words, the objects that impact traffic flow and their spatial features are studied in the model. The model also uses data from microscopic and macroscopic levels (Oberoi et al, 2017). A combination of qualitative and quantitative data helps to create a dynamic traffic model.

The analysis of various traffic models suggests that modelling is an important part of traffic management because it can evaluate increases in speed and reliability of changes in road capacity. Moreover, it can estimate impact of policies to reduce demand through various approaches. Modelling plays a crucial role in forecasting responses of road users and allows for superior traffic management. Congested networks can be upgraded to increase capacity that reduces road user travel costs. Improved routes or networks can be identified and developed through the use of modelling.

2.7.6 Anomaly Detection

Anomaly detection seeks to find erratic, peculiar, or exceptional data that do not comply with notions of normal behaviour. In other words, data that appears to be exceptions might represent problems in the entire system that must be identified and countered (Zhu et al, 2019). As indicated previously in section 2.7.2, road traffic anomaly detection is important for responding to congestion on road networks due to any factors. It is used to detect accidents, traffic controls, protests, disasters, and other incidents that might block or disrupt traffic flow. Additionally, the effective detection of road traffic anomalies provides road network managers with information that enhances their ability to deal with incidents in an effective and timely way (Demestichas et al, 2021). Anomaly detection addresses road traffic management challenges such as forecasting periods when roads will be congested. It can also be used to evaluate impact of accidents or extreme weather conditions.

The behaviour of road traffic is influenced by multiple factors, including the date, day, and time of day (Demestichas et al, 2021). Collisions and sudden events can be difficult or impossible to predict. Anomaly detection is done by processing, structuring, and analysing road traffic data. Individuals can study road traffic patterns, while anomalous events can be detected using machine learning techniques or statistical methods.

Among the different types of anomalies that can be detected are point anomalies. A point anomaly is defined as a single instance in a dataset that is differentiated from others when attributes are carefully studied by individuals (Xu et al, 2018). Contextual anomalies are data instances that are irregular or peculiar in a specific context (Xu et al, 2018). In other words, a contextual outlier has a different value from remaining data points in the same context. However, instance may not be identified as an anomaly if it occurs in a different context. Collective anomalies are data instances that appear to be anomalous with respect to the entire data set (Xu et al, 2018). The values of individual data are not anomalous in a contextual or global sense.

Traffic anomaly detection involves data collection and data processing. The data collection process starts with the use of detection and surveillance technologies that collect traffic flow data (Doshi & Yilmaz, 2020). Data processing involves the use of algorithms to detect and classify incidents through the analysis of traffic parameters. Observers can obtain information about the occurrence, severity, and location of traffic anomalies (Doshi & Yilmaz, 2020). Various researchers have proposed a number of traffic anomaly detection algorithms.

Li et al (2009) proposed an outlier detection algorithm by using a dynamic dimension which is time. The algorithm detects time related anomalies by evaluating historical trends between data points. The historical trends studied are related to each other. The drastic changes in trends are used to evaluate and detect traffic anomalies.

Djenouri et al (2019) studied various anomaly detection approaches for use in urban traffic analysis. They classified anomaly detection algorithms as being of either the flow outlier detection type or trajectory outlier detection type. Flow outlier detection algorithms use statistical, similarity and pattern mining approaches. In contrast, trajectory outlier detection algorithms use various trajectory detection approaches to help policy makers to solve or mitigate the problems.

Deublein et al (2013) proposed an anomaly detection approach that uses multiple techniques, such as regression analysis and Bayesian inference algorithms, to predict the occurrences of road anomalies. They also used wavelength transform methods to detect anomalies in road traffic (Deublein et al., 2013). This proposed approach to anomaly detection is important because data can be transformed into critical actionable information. It also provides useful insights for observers and professionals working with road traffic.

Santhosh et al (2020) proposed a vision-guided anomaly detection algorithm that language-based deep learning methods and relevant data sets to monitor the traffic on any road. The algorithm is useful because it detects different types of anomalies that policymakers can address to improve road safety.

Bai et al (2019) developed an anomaly detection system that consists of background modelling, perspective detection and spatial-temporal matrix discriminating modules. The use of multiple modules is important because road segmentation results and perspective maps can be obtained (Bai et al, 2019). Spatial-temporal information matrix helps to provide details about specific anomalies within the road system. The implications are that the model helps to provide data from multiple sources and can identify the most important data sets for analysis.

Feng et al (2021) developed a framework that uses multiple fibre modules to detect traffic anomalies. The network uses video streaming attributes such as group convolution and sparse connectivity to obtain real-time data and quickly extract anomalies (Feng et al, 2021). The benefits of the model are that it helps to enhance traffic anomaly detection in complex scenarios. The model's detection accuracy and recall rate were found to be better than those of other traffic anomaly algorithms.

Silva et al (2017) propose a traffic anomaly detection algorithm that uses smartphone sensors. Data mining algorithms were used to develop a system that can detect traffic anomalies. Although the system appears to be simple, it does have applications in detecting different types of anomalies. While Zhu et al (2018) employed a convolutional neural network (CNN) to detect traffic incidents automatically. The algorithm was found to have higher detection rates and lower false positive rates than other algorithms to which it was compared. The algorithm was expected to improve the accuracy of anomaly detection in large urban networks.

Thajchayapong et al (2012) developed a traffic anomaly algorithm that can be used as a distributed system. It utilizes microscopic traffic variables to identify and categorize traffic anomalies. The benefits of the algorithm are that anomalies can be identified and detected in various traffic conditions. An analysis of various anomaly detection algorithms conducted as part of the study revealed that each has its advantages and disadvantages.

Aboah (2021) applied machine learning approaches to the detection of the beginnings and endings of traffic anomalies using the NIVIDIA AI[2] smart city dataset, which is an open-source dataset containing videos of road traffic. The study used a total of 250 video frames, 100 of which were used for training and 150 for testing. A smart decision

[2] See https://www.nvidia.com/en-us/research/ai-demos/

tree model was used to train the system in background and foreground image detection. The results showed that the root mean square and accuracy of detection were 0.84 and 0.97, respectively The research work and results are readily understandable but have some limitations. For example, the feature extraction techniques and system specifications for training the model were not mentioned (Aboah,2021).

Nishant et al (2021) presented deep neural network techniques for congestion detection and estimation. Their paper presented a method for using fully connected layered neural network structures to estimate traffic congestion using high-resolution images. Although the algorithm yielded promising results, the use of layered structures makes the system cost-intensive.

A review of various anomaly detection algorithms reveals that each has its own set of advantages and disadvantages. The main advantage of anomaly detection algorithms is that they allow the main classification algorithms to work on the classification of objects of interest only. The anomaly detection algorithm picks up any abnormal event. Therefore, it is highly recommended to use anomaly detection algorithms in conjunction with classification algorithms. The main disadvantage of anomaly detection algorithms is that they are complex and often intimidating. They require extensive high-quality images to detect a particular anomaly. The presence of noise or lack of sufficient clarity diminishes the overall performance of the algorithm.

While numerous anomaly detection methods have been proposed, each uses its own data set and data type to achieve success. The literature also reveals that no one anomaly detection method is agreed to be the best and most efficient. In addition, there is no one standard technique for detecting anomalies. Most of the anomaly detection methods reviewed use image processing, which is expensive to use (Aboah, 2021; Nishant et al., 2021) as it requires high-quality images. Thus, any unclear image will affect the performance and accuracy of anomaly detection. Therefore, the work described in this thesis was conducted using datasets that do not rely on images, to simplify the learning process and improve the performance of the proposed framework.

2.7.7 Big Data Analytics for Intelligent Transport Systems

Big data analytics is defined as the application of advanced analytical techniques to evaluate and process large and diverse big data sets (Osman, 2019). It can analyse and process data belonging to different sizes and coming from multiple sources. Many industries and systems are using big data analytics for identifying trends, comparing data with historical data, increasing operational efficiency and making decisions based on real-time information (Osman, 2019). Big data analytics helps to make better informed decisions regarding strategies that can enhance and improve traffic conditions and safety. Moreover, it can result in process efficiencies and optimizations in the road traffic network.

Intelligent transport systems (ITS) are control and information systems that apply integrated communication and data processing technologies to improve mobility of individuals and goods (Perallos et al, 2015). Furthermore, ITS helps to improve safety and mitigate traffic congestion. The ITS uses innovative services related to transportation modes and traffic management. It has a traffic management centre that is the basic unit. The technical system collects and analyses data for operations and control management in real time.

The role of Big data analytics in ITS has been studied in the context of the proposed benefits that it would give to the latter. Specifically, Javed et al (2019) finds that big data analytics can assist in managing diverse and complex data. For instance, ITS can perform data storage, data analysis and data management effectively. The study cites big data platforms like Apache and Spark that have formidable data processing capabilities (Javed et al, 2019). Another study finds that big data can improve ITS operation efficiency (Torre et al, 2018). The study finds that ITS have sub-systems that manage large amount of data for information or facilitating decision making in traffic management (Torre et al, 2018). Big data provides timely data collection and analysis of current and historical traffic data. Consequently, management can identify riders' journey patterns in the transportation network. The long-term benefits for traffic department are superior public transportation service planning. Nuzzolo & Comi (2015) find that big data analytics can facilitate users by recognizing the most suitable route for destinations. Moreover, the shortest possible time can be given to motorists and passengers so that they can reach their destinations.

Research concludes that big data analytics will improve ITS safety levels by using sophisticated sensors and detection techniques (Chowdhury et al, 2017). Additionally, real time transportation information can be obtained that can predict traffic incidents or accidents. The implications are that big data analytics can facilitate ITS by improving the emergency rescue ability. Another study finds that big data analytics helps to identify assets problems like pavement degradation and ballast aging (Amini et al, 2017). Consequently, policy makers can make timely maintenance decisions that prevent infrastructure degradation. It can be concluded that big data analytics can enhance ITS in any urban setting. Moreover, big data analytics can provide optimized services and superior decision making. Traffic incidents and emergency response can be augmented through the application of big data analytics. It is understood that data plays an important role in smart or intelligent transport domain, therefore, a secondary contribution presented in this thesis is to study the available datasets that are used to evaluate transport machine learning-based solutions and identify the suitable set which is presented later on Chapter 4.

2.7.8 Traffic Congestion Worldwide and in the UK

Traffic congestion has worsened in recent years with the rapid rise in urban areas and human population. The large number of vehicles on roads has created a serious crisis that is being managed by using various strategies. Research finds that the top cities with traffic congestion problems in the world are Manila, Bogota, Mumbai and Pune in 2019 (Wang & Debbage, 2021). The cities have been ranked according to time when congestion is heaviest and lightest and comparison of highways with surface streets. Moreover, the number of hours wasted by motorists as they wait for other drives to get out was used in the study (Wang & Debbage, 2021). Research concludes that drivers in the United States lose 99 hours due to congestion (Dale et al, 2020). Moreover, Boston, Chicago, Philadelphia, New York City and Washington are the most severely congested cities in the United States. The findings also suggest that time lost in traffic caused losses of $88 billion to the economy in 2019 (Dale et al, 2020). The implications are that congestion causes extensive economic losses and inconvenience to commuters and motorists. Moreover, it also shows that lack of effective urban planning and management continues to cause extensive problems.

Traffic congestion is also a serious problem in the United Kingdom. According to research, an average British driver will spend 132 hours per year stuck in congestion

(Onyeneke et al, 2020). Moreover, traffic congestion caused £8 billion losses to the economy in 2019 (Onyeneke et al, 2020). London is cited with the second the worst traffic congestion in Britain. Londoners on an average spend six days and 5 hours in peak hour traffic. However, London’s mass public transit system provides a partial alternative for commuters that can avoid using vehicles for traveling to different places. In contrast, Edinburgh, the Scottish capital, is the most congested city in the UK. Drivers in Edinburgh spend 41% extra travel time stuck in traffic (Atta et al, 2020). Moreover, the city is the 33rd most congested city on the planet (Atta et al, 2020). The implications of the above research are that the UK needs to have an effective traffic congestion management strategy. Existing approaches have failed to stop the problem and prevent losses to the economy and inconvenience to motorists and passengers.

2.8 Discussion

Traffic congestion management applications face irregularities in their systems, which significantly hinder smart transport systems. One solution to these irregularities is the development of a sustainable and resilient transportation system that can detect anomalies in the application system. To improve overall transportation systems, identification and quantification are necessary, as they can inform the formulation of appropriate strategies. As previously discussed in Section 2.7.2, traffic congestion has caused great losses in the last few decades. Researchers and transportation manufacturers are working to reduce traffic congestion by detecting root causes that involve poor infrastructure, unpredictable weather, unknown abnormalities in the operating systems and the ageing of roads (Samal et al., 2021; Moyano et al., 2021). Different solutions to these hindrances in developing sustainable smart transport systems using IoT sensors have been proposed in the present era (see Section 2.3).

Various measures have been taken to address several performance criteria, such as speed, travel time delay and level of services (Falcocchio & Levinson, 2015).

In addition, the literature review also reveals that a fair amount of research has been directed at the mitigation of traffic road congestion, which was discussed in Section 2.7. However, there are certain drawbacks associated with the reviewed literature in these sections that must be addressed. Most of the reviewed literature was costly due to the nature of the machine intelligence approach, which either required very powerful machine models that deal with huge amounts of data (Kumar et al., 2018) or used computational resources excessively and consumed resources (as reported in Sections 2.7.3 and 2.7.6). In addition, the review revealed that a number of suggested solutions increased the complexity and execution time (Sun et al., 2019; Akhtar & Moridpour, 2021). As a result, these solutions can be unreliable and can also be expensive to adopt to detect and predict traffic congestion.

Another finding from the review in Section 2.7.3 was the lack of adaption to changing congestion levels over a period of time, as discussed in the works of Ranjan et al. (2020) and Chen et al. (2018). This can happen if new data attributes are used, which causes more time and resources to be consumed and consequently increases the complexity and execution time. This is one of the open issues that has not yet been addressed in the research but should be investigated, as one of the goals related to congestion detection is the identification of abnormalities as quickly as possible.

As reported in Section 2.5, an intelligent transportation system consists of many smart processes that adopt a modular approach but work in harmony with each other. Understanding vehicular traffic congestion is key to effective mobility and high-quality traffic management and safety systems. The resulting traffic congestion has a counter-effect on traffic flows in networks. A more empirical approach reveals that congestion on roads occurs due to sudden breakdowns. Vehicle speed decreases very fast, while vehicle density increases in what was initially a traffic-free road. There is a need for a traffic model that explains the empirical features of traffic breakdown and the resulting congestion. To explain this, this thesis aims to study and discuss traffic congestion issues using the HTM ML technique and to build on them to create a more accurate model that can detect abnormalities in the flows in real congested traffic.

In Section 2.6, the literature review revealed the importance of using machine learning in smart transport to detect and predict congestion. Furthermore, it is a potential solution that can decongest roads and is better to use due to its capabilities and advantages reported earlier in Section 2.4. However, even with the foreseen benefits of machine learning algorithms, the number of publications in this area remains limited, and new methods can be investigated to produce more reliable solutions for real-time applications, such as traffic prediction and detection. Such approaches, however, have failed to address traffic congestion anomaly detection and prediction in terms of dealing with huge amounts of online streaming data, as previously indicated in Section 2.7. Furthermore, the data generated by sensors in the IoT paradigm are huge, which further emphasises the need for a machine that is capable of dealing with huge amounts of data. Therefore, to address traffic congestion anomaly detection and prediction, an online algorithm is necessary to deal with it properly, which can be done using HTM. HTM is considered a new method, and it tries to realise the gap between neural mechanisms and intelligent methods (Price, 2011), which will be explained in detail in the next chapter.

In terms of the datasets, the literature discussed in Section 2.7 details the lack of a standard dataset for detecting abnormalities. Most of the reviewed publications used image processing (Aboah, 2021; Nishant et al.), which is expensive to use due to the fact that it requires high-quality images; thus, any unclear image will affect the performance and accuracy when detecting anomalies. Therefore, the work in this thesis will use datasets that do not rely on images to simplify the learning and subsequently improve the performance for the proposed framework.

2.8.1 Research Gap

Although traffic anomaly detection and prediction in the context of smart transportation have been studied in the literature, more research is required to make these applications more reliable and responsive to changes in road status. As discussed in Section 2.8, there are several challenges that need to be investigated, such as the high application cost of some due to their excessive resource utilisation (Akhtar & Moridpour, 2021), which may result in the model's inability to detect and predict anomalies in road traffic. Taking the challenges presented in Section 2.8 into consideration and examining the currently available literature, it becomes clear that there is a lack of well-studied anomaly detection and prediction in the smart transport domain.

In smart transport traffic congestion management applications, time is a crucial factor in anomaly detection and prediction. This means that detecting anomalies as early as possible must be considered when designing any solution. Therefore, a framework that is based on the HTM algorithm is proposed and implemented in this thesis to fulfil the gap in knowledge regarding congestion detection and prediction and to address the requirements related to them.

HTM has not yet been applied in smart transport congestion detections and predictions. Hence, more research is needed to study HTM to detect and predict anomalous behaviours in smart transport traffic congestion applications, which is presented In Chapter 3. In addition, traffic congestion management is a real time application. Therefore, it needs on-line algorithm to deal with it properly, HTM is an online-algorithm, and it learns from data stream in continuous way, whereas the other algorithms use batch data for learning (pre-trained).

As reported in this section, data play an important role in smart or intelligent transport domains. Additionally, one of the gaps found in the literature is the absence of a standard dataset that can be utilised as a starting point for anomaly detection and prediction in smart transport. Hence, a secondary contribution presented in this thesis is to study the available datasets that are used to evaluate transport machine learning–based solutions and identify the suitable set, which is presented in Chapter 4.

2.9 Summary

A comprehensive review of related published literature relevant to the research investigation was presented in this Chapter. The chapter first discussed the importance of AI and the goals of using AI, followed by the IoT definition and its major issues. The chapter also described the new ways of generating data for Intelligent transport systems. In addition, the literature review found that IoT and smart transport are growing fields. They also have a variety of potential solutions that would make roads less congested and better to use.

Then machine learning algorithms are reviewed to identify the current state of research and the research gap. This includes examining the concept, advantages, and limitations of machine learning models and how these models are assessed in Section 2.4. To investigate the role of Machine Learning in traffic congestion management the related work was reviewed in detail in section 2.5.

Furthermore, traffic congestion in smart transport systems is one of the most critical issues that need to be addressed. Therefore, the Chapter reviewed the related work in the context of smart transport, including publications on anomaly detection and prediction. As a result of the review, it was found that available solutions are in many cases not practicable or affordable (Section 2.7). Therefore, a more intelligent algorithm to detect and predict anomalies as quickly as possible is needed, so the thesis proposed the HTM framework, which will be discussed further in the next chapter.

The chapter concludes with a discussion of the research gaps in the literature for smart transport traffic congestion anomaly detection and prediction. The next Chapter will investigate the HTM as it will be adopted in the proposed framework.

Chapter 3 Hierarchical Temporal Memory

3.1 Introduction

This chapter discusses, the Hierarchical Temporal Memory (HTM) theory, which evaluates how the neocortex works and explains the neuroscience of the brain. HTM is a machine learning algorithm created in 2007 by Hawkins and George (Hawkins & George, 2007).

As an inspired biological technique built in the design of a neocortex, HTM attempts to mimic the process of how the human brain processes information. This processing occurs through vision, noise, character, and more, leading to memory and prediction. The HTM system is more trained than programmed, and the basic learning algorithms employed are not specified in certain sensory domains and may be used for a wide set of problems involving the modelling of complex sensory data(Chen et al., 2012; Walter et al., 2017).

The chapter is organised as follows. Section 3.2 describes the facts and background of HTM. Section 3.3 specifies the details of how the HTM functions. The section also reports examples of applications domains that applied HTM. Section 3.4 outlines the principles of HTM including the hierarchical structure of the HTM, the regions, and the sparse distribution representation. Section 3.5 shows examples of HTM applications, while Section 3.6 describes the characteristics of HTM that makes it a good choice for transport application. Section 3.7 describes the implementation of HTM theory, which is shown through the Cortical Learning Algorithms (CLA) and their components in details. The section also provides a detailed description of the encoders, spatial pooler, and temporal memory. Section 3.8 summarises the main highlights of the chapter.

3.2 Hierarchical Temporal Memory History and Crucial Facts

In 2004, Jeff Hawkins and Sandra Blakeslee introduced HTM, a new biology-based smart learning machine algorithm. This algorithm is based on neuroscience, which means that it attempts to mimic the functionality of the neocortex in the human brain (Hawkins & Blakeslee, 2004; Hawkins & George, 2006). Hawkins and George (2006) claim that HTM computational models can reach or exceed the performance level of the human brain for cognitive tasks. In addition, HTM can be useful in developing a machine capable of processing the large amount of data generated by sensors in the IoT paradigm. HTM is an inspired biological technique built in the design of a neocortex and attempts to mimic the process of how the human brain processes information through vision, noise, character, etc., thus leading to memory and prediction. The HTM system is more trained than programmed and the basic learning algorithms are used in it are not specified in certain sensory domains and may be used for a wide set of problems involving modelling complex sensory data(Chen et al., 2012).

The human brain neocortex consists of many types of neurons, with the most common biological type known as the pyramidal neuron(Hawkins & Ahmad, 2016). The neocortex inspires the design and functionality of HTM, which is projected to be used for sequence learning and making predictions. When fully operational, the algorithm should have the capability to produce general patterns for similar data inputs. When HTM is provided with time-series data in the experimental environment, it should use its knowledge to make time-reliant regressions. According to Sousa et al. (2021), performing such tasks would make the algorithm powerful in applications that utilise spatiotemporal data.

Figure 3-1 describes elements of the neurons of the human brain. The soma represents the cell body which is the first element of a neuron. The next elements, which are near the body are the dendrites, then the axon and finally, synapses. Bekkers(2011) revealed that the main task of pyramidal neurons is to convert synaptic input into a structured output of action prospects. Moreover, pyramidal neurons usually send their axons for long distances outside the brain, to control cognitive tasks. This movement allows the synapse cells to have either a strong or weak connection depending on the task performed over time, which is represented in the HTM hierarchical structure.

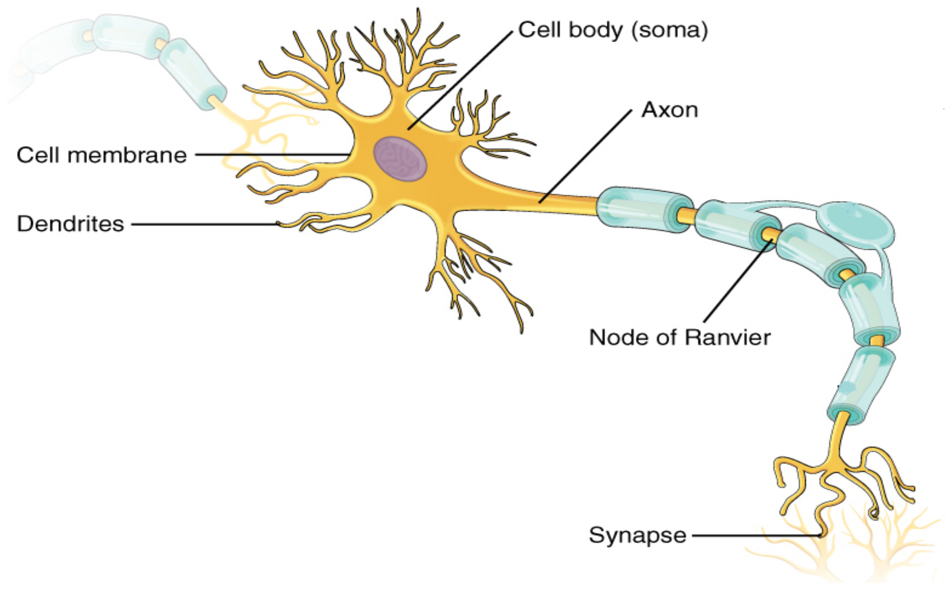


Figure 3-1: Basic neuron elements adapted from (Chruścik et al., 2021).

HTM has been on a long journey from its emergence to its present achievements. This progress is largely due to the persistence of Jeff Hawkins. As a neuroscience investigator, Hawkins is following up on his dream of building smart machines. He believes that the only way to build smart systems is to examine how the brain works, particularly its structure.

Chen et al. (2012) stated that Numenta has launched two generations of HTM algorithms. The first generation is known as "*The HTM Learning Algorithms*", which was proposed in 2007 by Deliep George and Robert 'Bobby' Jaros both members of Numenta (Hawkins et al., 2016). The algorithm was published by their company on March 1, 2007. Reflecting the basic idea of the official HTM, the algorithm follows the theoretical framework outlined by Jeff Hawkins in his book On Intelligence (Chen et al., 2012; Hawkins & Ahmad, 2016).

In fact, before the release of the first generation of HTM algorithm, considerable work had been done by research experts (Chen et al., 2012). This effort mainly includes the introduction of the foundation concepts and words behind HTM. In 2006, Jeff published a paper titled 'Hierarchical Temporal Memory: Concepts, Theory and Terminology', which was a precursor to first generation HTM algorithm (Chen et al., 2012; Hawkins & George, 2006).

The most important aspects of HTM are the time and hierarchy features. Figure 3-2 illustrates the HTM tree-shaped based on hierarchical levels. The levels are comprised of smaller components referred to as nodes or regions. One level can host several nodes. The number of nodes decreases in higher levels while there are more nodes on the lower levels. The higher levels function by relying on the patterns produced by learning lower levels. In addition, the higher levels memorize the prints to create even more complex patterns.

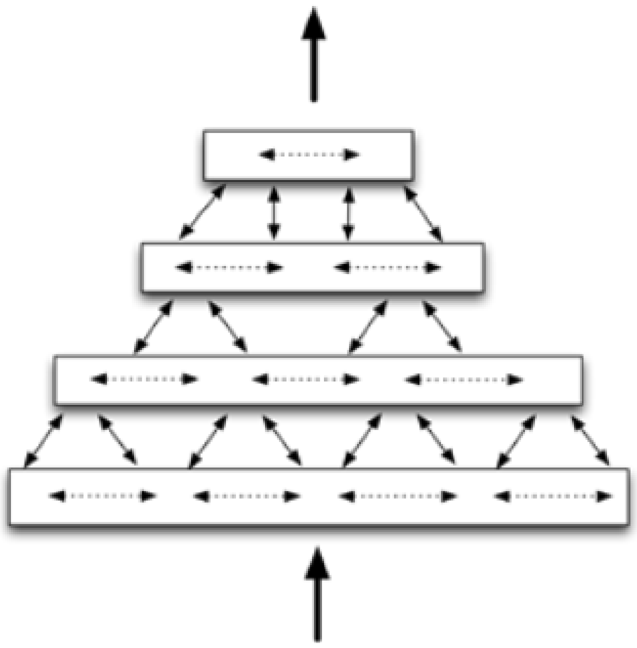


Figure 3-2: The hierarchical structure of HTM adopted from (Hawkins et al., 2010).

Time and hierarchy features in the vision problem represent the most crucial principles of HTM. Experts developed the idea by using the building neocortex and discovered that like humans, other mammals have a neocortex with a similar configuration and hierarchy. Moreover, humans and other mammals both see things from a single image without merging details many times (Chen et al. 2012). However, simultaneously managing the temporal and spatial states appears to be challenging. Moreover, the sequence of states, the static spatial configuration and reflection of things in time are problems that confused researchers. Eventually, a temporary situation caught their attention to aid in understanding and solving these problems. (Hawkins et al., 2016). This temporary information is used to find important aspects of general performance.

For instance, the problem of vision in mammals shows that mammals recognise items without supervision. In this case, time acts as a supervisor indicating which patterns match and which do not. In addition, although two images might be dissimilar, their simultaneous occurrence invites the conclusion that they are produced in the same timeframe. Hence, learning to identify objects comprises learning invariant representations, which is a concept initially introduced by Jeff Hawkins in his book on intelligence (Chen et al. 2012).

The first generation of HTM uses the identification of images as a sample to show the principle of HTM. In addition, an HTM network is a hierarchical computation whose variables are numbers of nodes. These nodes have similar algorithm, whereas each node has a child node and a parent node. Lower nodes are used to sense small range types of data whereas higher nodes receive the output of their child nodes and join them hence they sensor large types of data (Chen et al. 2012).

3.3 HTM working theory

The structure of the HTM network appears as a tree-shaped hierarchy, as seen in Figure 3-2 and Figure 3-3, consisting of levels that contains regions, each of which is composed of nodes (columns) that consist of cells. These components will be explained below in Section 3.4.2. . The top of the tree-shaped hierarchy has fewer nodes than the lower levels, which means less spatial resolvability.

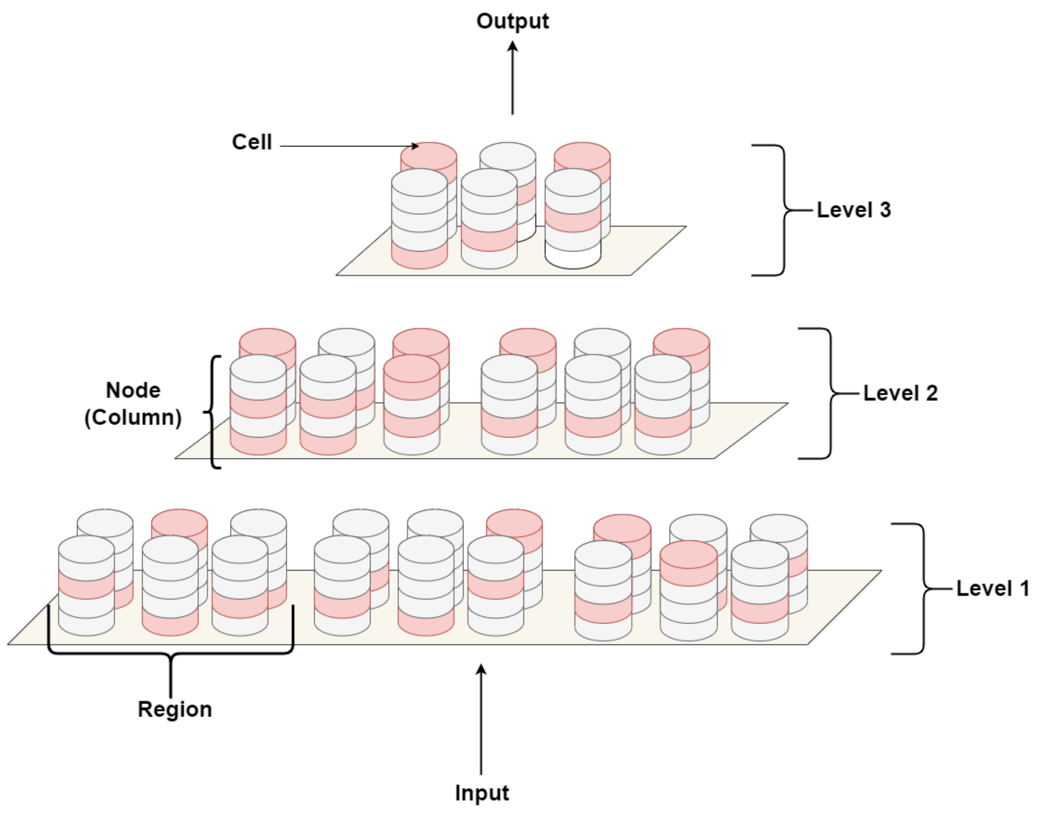


Figure 3-3: Hierarchical Temporal Memory structure.

The higher levels use patterns learned at lower levels to memorise complex patterns (Hawkins & George, 2006).

All nodes in HTM levels have the same functionality. The bottom level receives sensory data in learning and inference modes. After that, the bottom level generates a pattern in generation mode. The concept of the general category is usually stored in the top node, which is specified by smaller concepts in lower levels. More time and space restrictions occur in the lower levels than in the higher levels (Hawkins & George, 2006; Hawkins & Blakeslee, 2004; Garalevicius, 2007). Levels learn by identifying and memorising spatial patterns (e.g. inputs that happen simultaneously). Next, the HTM algorithm specifies temporal sequences of spatial patterns that are expected to happen consecutively (Garalevicius, 2007).

The two main modes are present in HTM algorithm: training and inference modes. The input of nodes during training mode is a temporal sequence of spatial patterns. In addition, the learning process of HTM has two steps: spatial pooling and temporal pooling (Garalevicius, 2007). On the one hand, ‘spatial pooling’ recognises patterns that are repeatedly observed, and these patterns are memorised as coincidences. To save memory, similar patterns are interpreted by HTM as the same coincidence. On the other hand, ‘temporal pooling’ divides expected consecutive events in the training sequence into temporal groups, each of which represents a so-called ’cause‘ (Hawkins & George, 2006; Garalevicius, 2007).

During the inference mode, nodes compute the possibility that a pattern belongs to a known coincidence and temporal group. It then assigns a set of probabilities to these groups that are known as node's belief (Garalevicius, 2007).

3.4 HTM Principles

In this section, principles of HTM are discussed. These principles include hierarchy, time, regions, and sparse distributed representations, which are described below.

3.4.1 Hierarchy

The design of an HTM network consists of hierarchically arranged regions. Each of the regions has temporal and spatial pooler memory with hundreds of cells arranged in columns. The regions also have elements that connect to other regions via elements located at various levels of the hierarchy. The connection allows for inputs to flow from regions located at the lower level of the hierarchy as a feed-forward input to regions in the higher levels of the hierarchy. In return, feedback stems from the higher levels of the hierarchy to the lower levels. To enhance the operational efficiency of HTM, memory is usually organised hierarchically, which helps reduce memory usage and training time.

3.4.2 Regions

The HTM regions are organised into independent cells which are arranged in columns and connected to one another. An example of this can be seen in Figure 3-4, which depicts an HTM region equivalent to a single neuron layer in the neocortical region. Input from the senses is usually received by the cells in the lower region. The output of the lower region is taken in as the input of the cells in the higher region. This creates a chain that facilitates the transfer of data from the lower levels of the hierarchy to the higher levels.

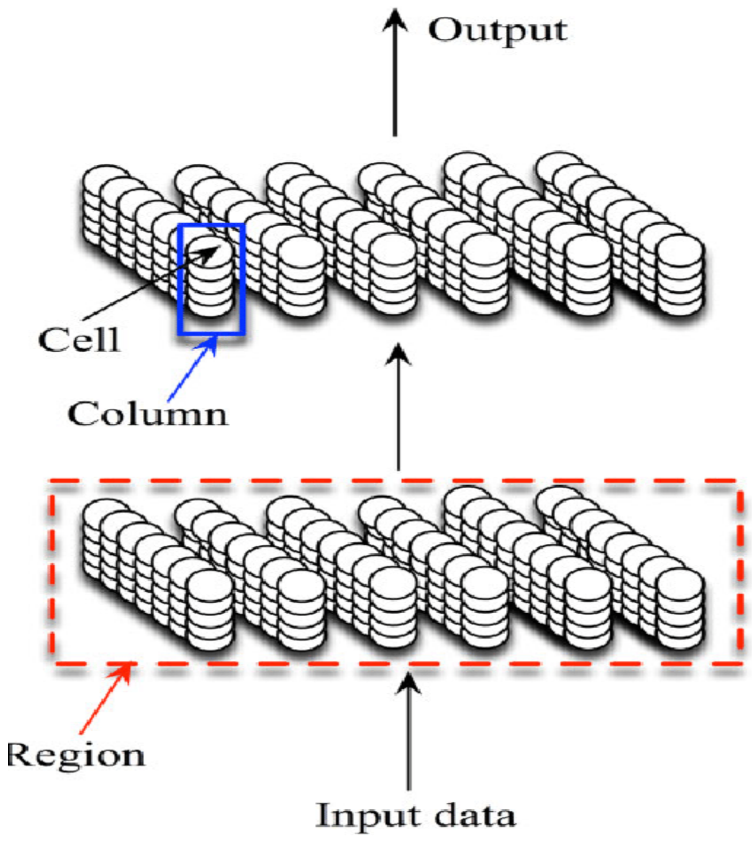


Figure 3-4: Region structure adopted from(Zhang et al., 2018).

3.4.3 Time

In HTM theory, time is essential in learning, as well as inference and prediction. In the context of the human brain, the importance of time appears when the brain must identify any object that comes from the sensors. The brain begins to identify the object by comparing the sensory object that flow to the brain during a period of time to identify that object. Hence, in HTM, identifying input patterns and learning their time is a significant aspect of training models (Hawkins et al., 2011).

3.4.4 Sparse Distributed Representations

At any given time, the regions are usually undergoing sparse active neuron formation with only 2% of neurons being in an active state. An example of sparse active neuron formation can be seen in figure 3-5. The representations or formations constitute the cornerstone ideas supporting the representation theory and are referred to as Sparse Distributed Representations (SDRs).

The neocortex has numerous neurons located in different regions. Each of the regions only have a limited number of active neurons at any point in time in a phenomenon that is referred to as Sparse Distributed Representation (SDR). SDRs usually consist of a specific number of active bits that represent a semantic meaning. In situations where two SDRs located in the same positions have active bits, the two SDRs tend to have similar semantic meanings (Hawkins et al., 2016). This phenomenon is referred to as 'SDR overlap'.

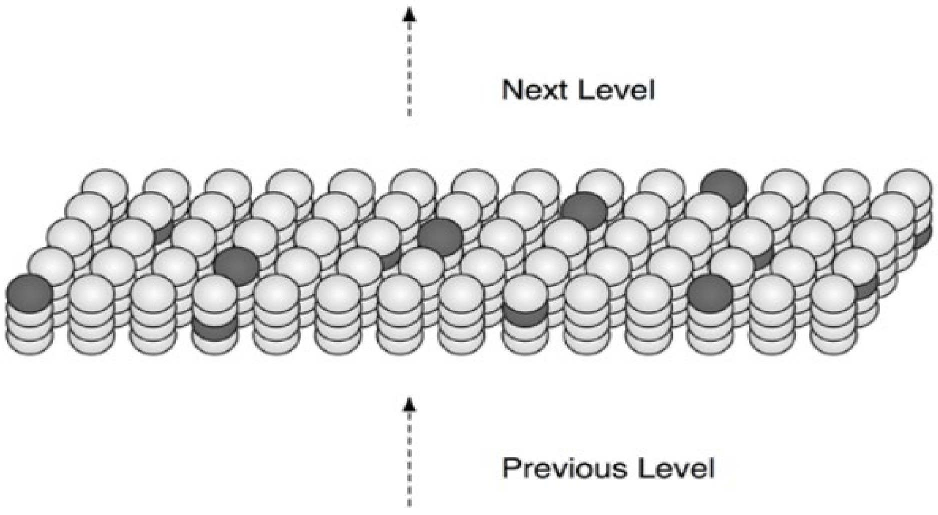


Figure 3-5: SDR representation in regions adopted from (Hawkins et al., 2010).

Unlike in ASCII encoding where a change in one-bit results in a change in the total meaning of the data, bit changes in SDR does not have an effect on the meaning of the data.

SDR size and sparsity are defined by the number of bits, which is denoted by the letter 'n', known as 'binary dimensional vector'. The width for one bit is denoted by the letter ‘w’ (Ahmad & Hawkins, 2015). The overlap for a two SDRs named a and b is calculated using equation 3.4-1. Moreover, when a two SDRs overlap reaches a threshold called θ, then the two SDRs are identified as matching SDRs, which is computed using formula 3.4-2.

$$overlap(a,b) \equiv a \cdot b \qquad 3.4\text{-}1$$

$$match(a,b) \equiv overlap(a,b) \geq \theta \qquad 3.4\text{-}2$$

For example, SDRs a *and* b with *n* = 40 (the number of bits), and *w* = 4 (total active bits) represented as zeros and ones.

$$a = [1000000000000000000100000000000110000000]$$

$$b = [0100000000000000000100000000000110000000]$$

In this example, the overlap is 3, hence, they can be matched if θ was equals to 3 (Ahmad & Hawkins, 2015).

The central data structure for the HTM system is SDR, , which is employed in various regions of the HTM hierarchy where it carries out various operations for the HTM system. Due to the central role SDR plays in HTM systems, all data fed into the HTM system should be encoded in the SDR format(Wu et al., 2018; Cui et al., 2016). The data conversion occurs at the bottom layer of the HTM hierarchy which has regions with the capacity to convert data into SDR format (Hawkins & Blakeslee, 2004). Once the data is successfully converted into SDR, all subsequent HTM outputs as one climbs up the hierarchy are in SDR format.

3.5 HTM Functions

Within the conventional HTM network, each region is responsible for executing only three core functions namely learning, inference and prediction. These functions are explained as follows.

3.5.1 Learning

Cortical HTM learning algorithms are online based thereby eliminating the need for separating the learning phase from the inference phase because the online learning capacity makes it possible for the regions to continuously learn from new data inputs. The learning process in the HTM region entails the identification of patterns in a given dataset. The region usually does not require clarification on the meaning of each data input because it evaluates data that is in SDR format. SDR formatted data consists of an array of binary numbers that constitute a mix of data bits that regularly occur together. HTM employs SDRs as tools for helping it learn about the spatial patterns of input data. With regards to learning about temporal patterns that reflect the formation of spatial patterns HTM tends to relay on time.

3.5.2 Inference

The learning phase is followed by the inference phase, the HTM matches the newly input information to spatial and temporal patterns already identified from previous data. This process occurs primarily because the inference stage is similar to pattern recognition, where the HTM attempts to identify patterns in the data.

During the inference phase, the HTM continuously evaluates data input streams with the aim of matching them to previously learned sequences. By using SDRs, HTM regions are able to handle the novel input in the inference phase. This is because SDRs have the capacity to match portions of a given dataset with confidence over the significance of the match.

3.5.3 Prediction

HTM regions perform predictions by matching new input with sequence of patterns and transitions that are stored in the regions. HTM regions are only capable of carrying out predictions if a majority of the HTM memory is allocated to the storage of transitions between patterns or sequence memory. Some of the major characteristics of HTM prediction include:

- Continuousness. HTMs operate in a state of constant real time prediction. Prediction is a central aspect of the HTM region because it is at the core of its operations. The characteristics and behaviours of the prediction and inference are almost the same, which makes it easy to identify them.
- Ubiquity. Prediction occurs in every region at every level of the hierarchy.
- Sensitivity. HTM regions usually employ both historical and current data in making predictions, which allows the HTM regions to make predictions that are context sensitive. A major feature of HTM regions is variable order memory which makes it possible for HTM regions to learn how to use contextualised data when making predictions.
- Stability. In a region, the prediction can be occurring in several steps to the next time point.

- Detection. HTM regions have the capacity to act as detectors that can predict expected operations. This capability makes it possible for the detectors to identify when anomalies occur.
- Evaluation. HTM prediction capability makes it possible for the system to evaluate a given data set and fill in sections with missing data. The system achieves this by utilising potential patterns that match with the given dataset.

3.6 Applications of HTM

Numenta[3] states that any application that uses data produced over time (e.g., GPS points and numbers) can suit HTM algorithms. Moreover, the algorithm will work with any application that has an inherent data structure (rather than being entirely random). The HTM algorithm is also suitable if multiple models are needed rather than one large model. Furthermore, HTM will fit any application whose main objective is to predict or detect anomalies(Numenta, 2005). HTM can be applied to a wide range of possible application fields including gaming, automotive, network modelling, drug discovery, vision systems, digital forensics, market analysis, business modelling, language processing, identification, and geospatial tracking(Hawkins & Blakeslee, 2004; Cui et al., 2016; Hawkins & Ahmad, 2016). As an example, Numenta published a white paper describing an application to detect patterns and anomalies in a vehicle's movement and speed, which is related to this research (Numenta, 2014).

One of the fundamental features of the HTM algorithm is that it considers the temporal context of data. Furthermore, information is stored throughout the hierarchy in HTM

[3] https://numenta.com/

algorithms, which learn continuously, eliminating the need for batch training. HTM algorithm also is an unsupervised learning, method, and it uses time as supervisor. Moreover, HTM algorithms can generate multiple simultaneous predictions, which makes it a good choice to apply to smart transport solutions(Hawkins & George, 2006; Cui et al., 2016; Hawkins & Ahmad, 2016).

3.7 Why Apply HTM to Smart Transport

As discussed earlier in this chapter, the three fundamental functions of HTM algorithms are learning, inference and prediction. The HTM model creates a model of its world, continuously learning complex patterns with the ability to recognise new patterns (Hawkins & George, 2006). HTM models can also adapt quickly to changing patterns in data streams, and learn multiple patterns simultaneously.

The main advantages of applying HTM to smart transport are its ability to generalise, robustness to noise, proficiency with online learning and its effective memory usage, which are explained in detail in the following section.

3.7.1 Generalisation

In HTM, ‘generalisation’ means that a single HTM model can handle different patterns (Wu et al., 2018; Fu et al., 2015). HTM applies the same selected parameters based on known features of actual cortical neurons and the same cortical region is used for various tasks. Therefore, no parameter adjustment is required. (Hawkins & Ahmad, 2016). As HTM is not a domain-specific learning algorithm, it can address a variety of scenarios identifying and defining the settings of one road to generalise and for use with other roads. This feature makes it easy to use in any transport scenario.

3.7.2 Robustness to noise

For time series-based anomaly detection in particular, an important key element of HTM algorithms is the use of SDRs (Fu et al., 2015; Hawkins & Ahmad, 2016). With SDRs, the activities of the neurons are biologically inspired by the neocortex. The main task of the SDRs is to encode data. The ability to manage noisy data is one of the most useful characteristics of SDRs, ensuring that the temporal memory obtains the same output every time even if the data are noisy or corrupted. In some cases, a significant number of columns and cells may be lost in the SDRs, yet they are still able to make accurate predictions.

In practice, data from sensor devices may contain missing values or suffer from a significant amount of noise, including wrongly added labels or missing attributes due to various factors such as sensor failures or damage, weather factors, disconnections or equipment changes (Wu et al., 2022; Al Najada & Mahgoub, 2016). As highlighted in Sections 2.7.7 and 2.8 in Chapter 2, poor quality noisy data in smart transport systems can affect performance in detecting and predicting anomalies. HTM's robustness to noise feature allows the algorithm to handle noisy or missing data entry, which was one of the challenges that was identified in the research gap. Therefore, using HTM in this thesis could improve performance for detecting anomalies in traffic data.

3.7.3 Online learning

HTM is identified as an online learning model that can predict and learn future events in one or more steps. This algorithm works well with data that changes continuously over time. However, there is no need for traditional training or testing of the datasets to achieve high accuracy rates (Hawkins & George, 2006). More sensors are being added to smart transportation road infrastructure. Every sensor produces an extensive amount of data, which demands an intelligent algorithm that can manage frequent and constant data changes. There is no need to keep all of the newly streamed data because HTM is a memory system that can learn and forecast.

3.7.4 Effective memory usage

The memory requirements for HTM algorithms are less than other machine learning algorithms such asANN, deep learning,and others(Chen Xi et al., 2012; Wu et al., 2018). Due to the nature of the smart transport data, which are indeed very huge and updated every time, it requires a machine intelligence algorithm that use the resources efficiently. Furthermore, there is no need for batch learning with HTM algorithms (Cui et al., 2016; Hawkins & Ahmad, 2016).

3.8 Cortical Learning Algorithms (CLA) and its Components

Considering HTM is a memory-based system, it is very different from programming standard machines since it is training by a stream of sensory data. Furthermore, it is trained several times from different sets of data, storing a huge number of patterns and events. In addition, storing and accessing data in HTM differ from current standard machines. Because traditional computer memory has a basic system that would allow it to handle any data layout, developers may choose how data bits are saved. (Hawkins et al., 2010).

The implementation of the HTM theory is known as the cortical learning algorithm (CLA) that tries to describe the structural and algorithmic properties of the neocortex

(Hawkins & Ahmad, 2016). Moreover, CLAs models are examined and implemented in software produced by Numenta.

In the CLA, data streams are analysed, classified then it learns to detect abnormalities and then it uses patterns based on time to perform predictions tasks.

Furthermore, according to (Hawkins et al., 2010), in most animals the usual necessary actions that include hearing, touch, movement, and vision are all together managed by the neocortex as well as in humans, these skills are also accomplished by the neocortex. In Addition, HTM is a technique based on how the neocortex carries out these tasks. For most cognitive tasks, HTM has the potential of creating machines that will be equal to and even better than humans. HTM represents neurons as hierarchal layers of columns and regions.

The working concept of HTM is defined by allowing users to define the hierarchal size and type of training on the HTM network. Nevertheless, HTM does not allow the users to controls how to keeps the content (Hawkins et al., 2010).

The overall structure of the HTM is organised in a hierarchy that represents time composed of regions that represent the primary memory cells and feedback connections. This feedback connections between regions allowed the hierarchy to deviate as a whole descends the hierarchy. In top of that, time plays a significant part in HTM's learning, inferencing, and prediction processes.

In the CLA algorithm, which is the implementation of HTM, although the CLAs algorithms can learn from the input data stream to identify the temporal sequences in the data, it is still impossible to predict future patterns that may appears. The importance of this algorithm comes from its ability to handle building blocks of neurons that form the neocortex (Hawkins et al., 2010).

One of the most essential features of CLAs is learning from each cell in the input space associated with each column. Moreover, columns denote the semantic definition, and every cell in that column indicates the same meaning yet in a different setting (Agrawal & Franklin, 2014; Hawkins et al., 2010).

Originally, the CLA is runs a single region only. In CLAs, the significant elements in every region are the spatial pooler and the temporal memory, that is intends to describe the neocortex's structural and algorithmic properties. (Hawkins & Ahmad, 2016).

Figure 3-6 shows the components of CLAs, which are: spatial pooler, encoder, temporal pooler, and classifier.

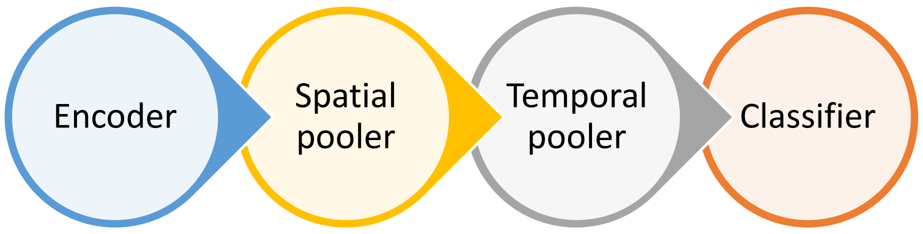


Figure 3-6: CLA components.

The encoder is the first component in CLA, which can be one or more than one encoder depending on the input streams. The function of the encoder is to change the inputs to an SDR. The second element in CLA is the spatial pooler, which facilitates conversion of the SDRs by the encoder. The main purpose of the spatial pooler is to identify the spatial patterns of the given SDR. The pooler then produces another SDR, which will be then passed on as an output to the following CLA part. Temporal memory is the next component in the CLA, which is mainly focused on learning temporal variations in the given SDR. The last CLA element is the CLA classifier, which aims to predict and classify tasks by decoding the CLA model state and producing predictions (Hawkins et al., 2017). In the following section, the CLA components are explained in detail.

3.8.1 Encoder

In CLAs, the encoder takes the data, turns them to SDRs, then passed them to the HTM. This step includes transforming the outputs into a serial of zeros and ones (Purdy, 2016b). Furthermore, the type of data input determines which encoder to use. The same data requirement output overlapping SDRs if the sensory input has the same semantic purpose (Purdy, 2016b).

Moreover, (Purdy, 2016b) summarised the important of the encoder as follows. Encoders can identify and link the same semantic data by overlapping data with inputs that share one or more features. In HTM, if two SDRs contains a significance overlapping bits, then it considered to have the same value.

Moreover, encoders are responsible for generating exact output for the given input. Encoders must also have the same dimensions to guarantee that the SDR can compare the bits and associate them together. Encoders should always create constant sparsity output. There are several encoders produced by NuPIC, such as scaler encoder, date encoder and category encoder, ...etc.

3.8.1.1 **Numerical Encoders**

Several encoders are produced by NuPIC to encode numbers. An example of NuPIC encoder is the scalar encoder that can encodes numerical data. Furthermore, integers and float data types are both be encoded using the scalar encoder. The scalar encoder comprises a range of interfaces that determining the overall number of bits and keeps the necessary encoder features in each of them. In NuPIC, the parameters that are included in the scalar encoder are: *minimum value, maximum value, number of buckets*, which is the number of all bits in the SDRs that will be produced by the encoder, and *w* which is the number of active bits.

3.8.1.2 **Date and time encoder**

Another example of NuPIC encoders is the date encoder. Unlike the scalar encoder, this encoder presents a more useful interface to deal with dates values in the inputs. The date encoder has many parameters to use in more detail, which includes *timeOfday* to work with hours, *DayOfWeek* to identify the days of the week as a numerical notation starting with zero representing Monday to six representing Sunday. Moreover, the date encoder offers more choices to identify the day type of the week into weekend or working days as a notation of boolean values (true or false). The parameters of the date encoder according to Numenta[4] are:

- timeOfday: represents the time during the day in hours.
- dayOfWeek: represents the value of a specific day.

[4] https://nupic.docs.numenta.org/1.0.3/api/algorithms/encoders.html#date-encoder

- weekend: represents if a day is weekend or not.
- holiday: represents if the date is a holiday or not.
- season: represents the season (spring, autumn, winter, summer) in the year.

This classification is a vital aspect when dealing with congestion as they tend to appear more likely at the weekends.

3.8.2 Spatial Pooler

The primary objective of the Spatial Pooler (SP) algorithm is the conversion of HTM region input into sparse pattern that can be used by the HTM system to learn about its sequences and make predictions. During the conversion process the SP, learns about the connections that exists between the columns and the input subsets, determines the level of input associated with each column and selects a sparse set of active columns through inhibition.

SP input consists of binary vectors of ’0‘ and ’1‘ with each column connected to a random subset of the input bits, referred to as the potential pool. The column and the input are connected via a synapse and each of the synapses has a permanence (value associated with the synapse). The permanence is a scaler value that ranges from ‘0.0’ to ‘1.0’. Learning at the column level usually happens through a systematic increase and decrease of the permanence. When the permanence of a synapse is above the threshold it is assigned a weight of 1 which causes the synapse to be affected by the input bit. In the event that the synapse permanence is below the threshold, it is assigned a weight of 0 causing it not to be affected by the input. When a new record is entered into the HTM region two processes occur namely the column selection process and the cell selection process. The column selection process entails selecting which of the columns will be activated. The cell selection process entails selecting which of the cells within the activated columns will be activated. This is important as each cell represents a different temporal context. The general rule is that the most current input is represented by active columns, thereby helping to ensure there is similar representations for similar inputs that are in a SP. The steps of the SP algorithm include parameter initialization, score overlap, inhibition, and learning. These steps are illustrated in Figure 3-7 and explained in the following sections.

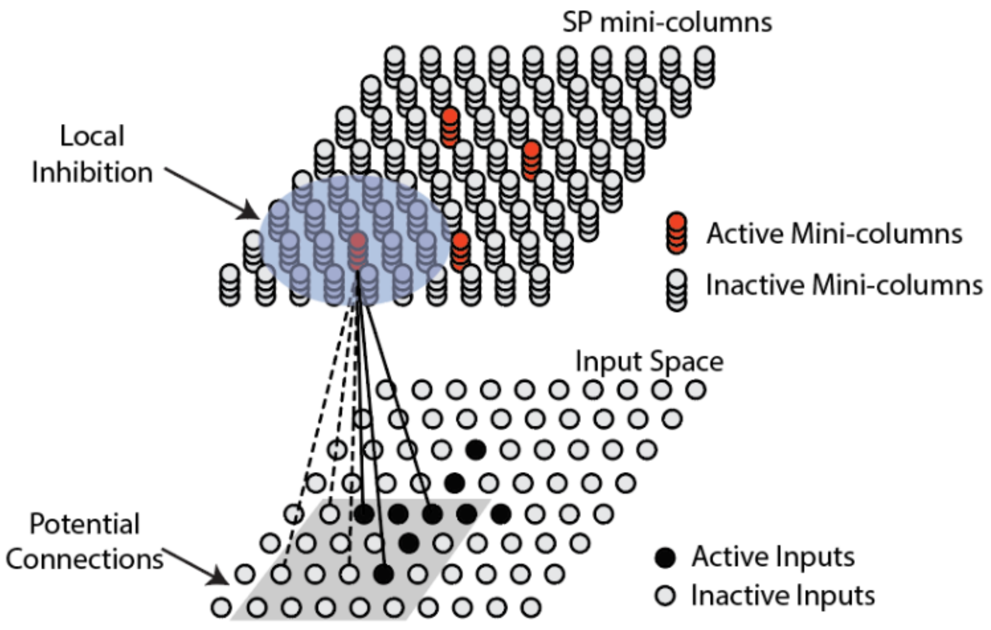


Figure 3-7: Spatial pooler process adopted from (Cui et al., 2017).

3.8.2.1 **Parameter Initialisation**

The initialization of parameters is the first step of the SP algorithm. Before receiving any input, the algorithm initializes its parameters. This is achieved by allocating a list potential synapse to each column. Each column is then linked to a random set of binary input that stem from the input space. Each of the input has its own synapses with assigned random permanence value. To allow for that the potential synapse is in a "*connected*" or "*disconnected*" state, the permanence value is in a small range around the permanence threshold.

3.8.2.2 **Overlap**

In the overlap score phase, the SP utilizes the provided input vector to calculate the overlap score for each column. The overlap score is calculated by summing to the number of synapses that are in a "connected" state and are connected to active inputs ('ON' bits), multiplied by the boost factor.

3.8.2.3 **Inhibition**

In this phase, the algorithm computes column inhibition. It also computes the winning column they evaluating which columns remain winners after the inhibition process. The inhibition method is defined using the "*globalInhibition*" parameter. The parameter covers both local and global inhibitions. The other two parameters that are used in the inhibition phase are the "*LocalAreaDensity*" parameter and the "*numActiveColumnsPerInhArea*". The "LocalAreaDensity" parameter is responsible for specifying active column density during the inhibition process. Meanwhile, the "*numActiveColumnsPerInhArea*" parameter is employed in the calculation of the maximum number of columns that remain active within the local inhibition area.

The inhibition step is crucial because it ensures that each of the columns has sufficient connections to the input bits to become active. For a column to be considered for activation during the inhibition process, its overall score has to be equal or greater than the "*stimulusThreshold*" parameter. This threshold is a number specifying the minimum number of synapses that have to be "ON" for a column. This is done to prevent noise input from activating columns, which is essential to obtain an accurate performance. In addition, in NuPIC the *"stimulusThreshold"* parameter is specified as a percentage of a fully developed synapse, and it is set to be "0" (Bastaki, 2019).

Furthermore, columns with a minimal number of connections usually do not meet the threshold for activation even if all the input bits have been connected for the activation process. For such columns to be activated the permanence value needs to be increased until they meet the required minimum number of connections for activation.

3.8.2.4 **Learning**

The learning phase only impacts the permanence values of the winning column. The synapses of the winning columns, their inhibition radius and boots value is updated as a result of the learning process. In the event that the synapse is active, the winning columns' permanence value is increased and if the synapse is inactive the permanence value is decreased. The learning phase is also the phase where the boosting is computed through one of two mechanisms that are designed to help columns in learning the connections.

3.8.3 Temporal Memory

The Temporal memory (TM) algorithm learns about pattern transitions by evaluating SP sequences over time. SP output is in SDR format representing the active columns of cells which received most input. The TM algorithm resolution produces representations of temporal sequences at the individual cell level within the columns that are active.

The TM algorithm crates a second form of SDR by activating a subset of cells located within the winning columns. The activation makes it possible for a HTM region to represent the same input in a variety of different contexts (Ahmad & Lewis, 2017). The learning phase of the TM algorithm is similar to that of the SP algorithm given that they both entail the establishment of connections through increasing and decreasing permanence values of potential synapses located on a dendrite segment.

Active cells in TM algorithm form connections with other active cells that had been activated prior to their distal synapses. This makes it possible for the cells to predict the moment when they became active by evaluating their connections. This feature makes it possible for the cells to store and recall sequences. The cells are also able to predict what will likely happen next by evaluating the sequences that they have stored (Cui et al., 2016).

3.8.4 Classifier

According to Hawkins et al. (2016), the purpose of the classifier is to decode the outcome of the model. After decoding, the classifier can predict based on the decoded outcome. One of the NuPIC' s available for detection and predictive tasks is the SDR classifier, the CLA classifier, and the KNN classifier. The SDR Classifier derives from the last CLA Category and accepts SDR outputs, also known as the activation pattern, from the TM algorithm and the encoder information. In other words, it transforms active SDR patterns into distributions of probability (temporary cell vector memory activation) (for the possible encoder buckets). This is done with one layer of feedback network by the SDR classification.

In terms of performance, the CLA is dominating the current reconstruction algorithm. The classifier collects the information from the TM method and coders (a classifier combines two or more of these components into a binary scheme). Temporal memory accounts for all time-dependent processes. This temporal trend classifier does not forecast the next shift; instead, it takes a comparison value and computes how the future could be similar to the present. If temporal memory is believed to include all the information stored at any given time, it is said to include everything from history to the way things are now.

3.8.5 Anomaly Detection

In HTM, anomaly detection is performed by calculating the anomaly score. The anomaly score constitutes the metrics that determines the predictability of each record. The anomaly score for each record ranges either 0 or 1, where 0 represents a value that is predictable, 1 represents a value that is anomalous. The core TM and SP algorithms constitute the platforms on which the anomaly score is usually implemented in HTM because they do not require any changes to the algorithm. A streaming anomaly detection system can be created using HTM anomaly detection. This is because it is open source, commercially deployable and performs quite well across an extensive range of data sources. HTM anomaly detection can also be deployed in the real-time health monitoring systems, monitoring of IT infrastructure, energy consumption monitoring and vehicle tracking.

3.9 Summary

This chapter discussed the HTM theory and its CLA computational implementation to get better understanding of how it works and to show the benefits of using it in any domine.

HTM memory is a time-based structure, which allowed it to reduce training time as one of its many advantages.

The chapter also included the CLA implementation of the HTM theory and its components. These components are the encoder which converted the input data streams into SDRs. Then it will be received by the Spatial Pooler from the encoder that learns the spatial features of the received SDRs. After that, the Temporal Memory can learn and understand the temporal changes in the SDRs. For prediction models the Classifier decodes the state of the CLA to be able to perform predictions.

Moreover, HTMs are robust to noise in data. In addition, traffic congestion management is a real time application. Therefore, it needs on-line algorithm to deal with it properly, HTMs are an online algorithm and they learn from data stream in continuous way, whereas the other use batch data for learning (pre-trained).

Similar to any machine learning algorithms the HTM models need labelled data to be trained from then tested and validated, which will be discussed in the next chapter.

Chapter 4 Datasets

4.1 Introduction

The key element in any machine learning (ML) model is the training and testing dataset. One of the objectives of this thesis was to identify the best dataset that can be used to evaluate the potential of ML techniques and algorithms in different/various smart transport scenarios. This objective has been achieved by following review criteria that focuses on (1) identifying previous datasets which relate to vehicle and road traffic in particular, (2) screening the identified datasets to exclude other traffic management data, such as those which relate to network traffic, air traffic, and railway traffic, and (3) including the remaining identified datasets that are available for use.

As indicated previously in Chapter 2, datasets are crucial for training algorithms in machine learning, as the performance of the trained algorithm depends on the suitability of the datasets. Therefore, one of the secondary contributions to knowledge in this thesis is to investigate and review the available datasets to identify the suitable datasets that can be used to test and evaluates the proposed framework. The first step was to identify the sources that generates datasets, which is one of the objectives of this project. The sources of dataset are shown in Figure 4-1.

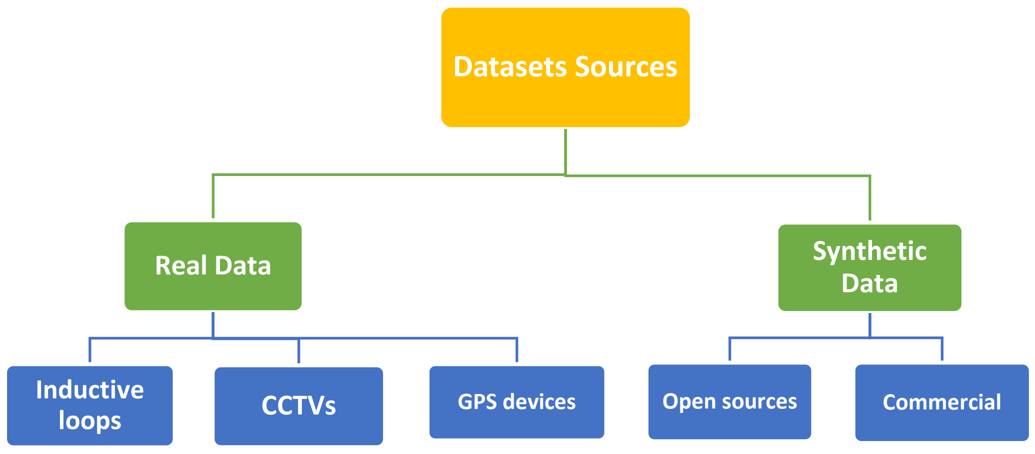


Figure 4-1: Datasets sources summary.

The sources of datasets (Figure 4-1) can be either real public domain data or synthetic data. One of the important issues facing smart transport research is the lack of representative and standardised datasets for anomaly detection tasks. Therefore, this thesis has carefully considered the suitability of existing datasets according to the following criteria:

1. Availability and accessibility: The datasets must be available to download and view

2. Reliability: This thesis investigated the work published over the past 10 years, as old data sometimes has lost too much of the contents due to the way it was obtained, as well as the type of sensor it was generated from, which in some cases is not as reliable to use.

3. Relevance: The datasets should be related to vehicle and road traffic, in particular, and not to other contents including air traffic, railway traffic, and other network traffic.

The outcome of this review based on the stated conditions are presented in the section 4.3. The rest of the chapter is organised as follows. Section 4.2 describes the data mining steps. Section 4.3 specifies the datasets review based on the selected criteria presented in section 4.1. The outcome of the review is to select a suitable data to be used in this thesis, which is presented in section 4.4. Finally, section 4.5 summarises the main highlights of the chapter.

4.2 Data Mining Processing Steps

4.2.1 Collection

Data is collected using the various methods and platforms as discussed before. Information and data like videos, images or hybrid methods ensure the collection and availability of data used in programming and instructing the traffic systems (Wang et al., 2017). These are collected using cameras, speed guns, state-pf-the-art machines for imaging and capturing fast moving objects.

4.2.2 Processing

Most of the evidence, data, and information collected are sometimes subjected to inconsistencies like missing values, noise, or poor-quality data due to sensor failures; data link errors, or even measurement failures and inaccuracies. As a result, data processing and manipulation are necessary for ensuring accurate, clear, and precise information is available. Data processing is done through data cleaning where malfunctions are detected, missing data is recovered, and noise is removed (Al Najada & Mahgoub, 2016). Data processing also involves data dimensionality reduction using kernel dimension reduction, non-negative matrix factorization or using manifold learning. By reducing dimensional spaces, learning-driven tasks are tremendously improved. Data processing also involves sparsity analysis, removing redundant factors and features using heterogeneous learning or compressive sensing. Finally, data processing can be done using data fusion, where data from various sources is processed.

4.2.3 Analysis

Analysis tools are used to understand traffic data which goes a long way in improving the existing situation or resolving a problem, such as estimating the number of vehicles in a roadway or directing traffic to other routes in case of accidents. Quality and accurate data analysis are essential in identifying erroneous data, measurements, and information affecting data-driven traffic practices and processes. Analysis of big traffic data is also necessary for making accurate and real-time traffic decisions and initiatives.

4.2.4 Storage

The ever extensive and ever-growing traffic data always raises the demand for larger and cost-effective storage facilities, hardware, and technologies. Because traffic is dynamic and always on the move, appropriate and secure storage facilities are necessary for making confidential traffic decisions and initiatives. After data has been collected, processed, and analysed, it is stored in technologies like cloud storage for effective and efficient real-time traffic initiatives and decisions (Yan et al., 2017).

4.2.5 Communication

After traffic data has undergone the stages mentioned earlier and phases, data communication/sharing of traffic data is conducted. Communication and sharing of traffic data play a pivotal role in the studying, constructing, designing, planning, monitoring, and operating traffic congestion management structures and systems (Nguyen et al.,2016). Such data is communicated to stakeholders like the government, urban planners, policymakers, researchers, and the transport department to make their systems cost-effective and more efficient. In this phase, traffic data sharing is done under ethical principles and concepts like security, privacy, transparency, and liability.

4.2.6 Maintenance and archiving

In this phase, data maintenance involves systematically checking and improving data through verifications, enhancements, and corrections. Through maintenance, data is preserved and stored in functional states that are utilizable and useful. When archiving, data that are less commonly used or are no longer helpful in active systems are moved and stored in unique archiving structures, systems, and technologies (Hung et al., 2016). Archiving is done to clear storage and space, optimize the performance of active systems, and allows for cost-effective storage that allows for future retrieval and utility.

4.3 Datasets Literature Review

The lack of representative and standardised datasets for anomaly detection tasks is one of the key issues affecting smart transport research (Hu et al., 2022).

As a result, the suitability of existing datasets has been carefully considered in this thesis based on to the availability of the dataset, which means that the datasets can be available to download and view. Another consideration is the reliability of these data, which can be determined by reviewing published journal articles over the last ten years.

This is because old data has lost too much of its content as a result of how it was obtained, as well as the type of sensor it was generated from, which is not reliable to use. Finally, the datasets should be specifically related to vehicle and road traffic and not to other contents such as air traffic, railway traffic, or other network traffic.

Table 4-1 summarises the outcome of dataset investigations over the course of this PhD research. The table displays the common attributes that have been applied to various machine learning algorithms such as ANN, GA, PSO, deep learning, and Fuzzy NN, and more. These attributes include the time stamp, speed, density, and GPS location.

One of the recent studies used real traffic data from loops detectors on the roads that is available from the Highways England's Motorway Incident Detection and Automatic Signalling (MIDAS) system which is fed to an ANN algorithm to predict the traffic status for the following 15 minutes of the journey (Goves et al., 2016b). In addition John et al.(2016) employed data collected by loop detector sensors from England's Highway Agency to introduce a framework to predict the traffic congestion every 15 minutes. This study used the speed and the density as an input to a Neuro Fuzzy Inference System (ANFIS), to provide the Level of Congestion for the future 15 minutes. Furthermore, Zaki & Ali-Eldin(2018), implemented a traffic prediction model using datasets that were collected using traffic cameras.

Another type of datasets was used is the trajectory data of the vehicles to estimate and predict the urban traffic congestion(Kong et al., 2016). Furthermore, Kumar et al. (2013a, 2015) used video cameras at selected locations for a period of five days from Monday to Friday from the National Highway of India at two selected locations to fed to ANN to predict the traffic.

A study by Chang et al.(2012) used the KNN algorithm to implement a dynamic multi-interval traffic volume prediction model using traffic demand data at the tollgate,. Data was automatically collected by the toll collection system and managed by the Centre for Operations Analysis and Supportive Information System, an ADMS at Korea Expressway Corporation (Chang et al., 2012). Another study by Kwoczek et al.(2014) used a combination of various sources to predict traffic in the presence of special events. The sources include floating car data, GSM data and data from a stationary sensor obtained from local traffic management centres in the city of Cologne. A data provided by the SCATS system (Sydney Coordinated Adaptive Traffic System) was used to detect traffic (Liang et al., 2018).

In addition, Illinois Department of Transportation provides datasets that was used in (Polson & Sokolov, 2017) study to predict traffic using deep learning module. The data was generated from 21 loop-detectors installed on a north bound section of Interstate I-55 in the US.

In (Julio et al., 2016)study, the data was from Tran Santiago, the public transportation system from Santiago, Chile. This study used speed and GPS location in a specific time to predict bus travel speeds using traffic shockwaves and Artificial Neural Networks (ANN), Support Vector Machines (SVM) and Bayes Networks (BN).

In a study by Tang et al. (2017), researchers aim to predict traffic speed over a period of two minutes using their proposed model: the evolving fuzzy neural network (EFNN) model. They collected data from three remote traffic microwave sensors that were located on the fourth ring road in the city of Beijing. The sensors were placed on a road segment that is approximately 2.74 km long. This segment is particularly crowded during rush hours. The performance of the EFNN model was associated with a lesser rate of errors in comparison to traditional models. Tang et al. allude that reaching at more accurate traffic predictions is crucial for advanced traffic management systems (TMSs) and intelligent transportation.

Another source of data that was applied to predict the traffic condition is using data generated by simulation tools. An example of using simulated data is the comparative analysis of machine learning algorithms to urban traffic prediction conducted by (Lee & Min, 2017) using SUMO (Simulation of Urban Mobility) tool to produce the datasets that used in their study.

Another example is the data used in (Jr et al., 2017) to generate the data using SUMO to detect traffic using deep learning. In addition, (Sturari et al., 2016) used simulated data generated by SUMO to design an integrated mobility system with traffic simulation capabilities, based on integration of many different sources of real-time traffic data. However, this is one of the limitations or drawbacks of the solutions since simulated data were not tested in real world, so the given results were not reliable according the authors of this study (Lee & Min, 2017).

On the other hand, there are studies that used both real and simulated data depending on the aim of the study and the type of the data that was needed to be used in the module proposed in these studies. Such as example of this case is the study conducted by (Tang et al., 2016) to build an accurate estimation time using datasets collected from loop detector. Furthermore, the study also used syntenic data that were generated using Vissim software. As for the real dataset, the study used DRIVENET (Digital Roadway

Interactive Visualization and Evaluation Network) dataset(Tang et al., 2016).

Table 4-1: Summary of reviewed datasets.

Reference	**Algorithm**	**Application Type**	**Data Sources**	**Used Data Attributes**								**Dataset's Name**	**Open Source**
				Speed	***Volume***	***Lane***	***Density***	***GPS***	***Time stamp***	***Vehicle type***	***Vehicles counts***		
(Goves et al., 2016a)	ANN.	Prediction.	Inductive loops.	✓	✓	✓	✓	✓	✓			MIDAS	Yes
(Kumar et al., 2013)	ANN.	prediction	video cameras.	✓	✓	✓	✓	✓	✓	✓	✓	National highways in north India	No
(Kumar et al., 2015)	ANN.	Prediction.	Same data in (Kumar et al., 2013) study.	✓	✓	✓	✓	✓	✓	✓	✓	National highways in north India	No
(Chang et al., 2012)	KNN-NPR	Prediction.	Tollgate traffic data.		✓			✓	✓		✓	Korea Express way Corporation	No
(Lee & Min, 2017)	RF, GB, KNN, MLP	Prediction.	SUMO.	✓				✓	✓			A map	No

Reference	**Algorithm**	**Application Type**	**Data Sources**	**Used Data Attributes**								**Dataset's Name**	**Open Source**
				Speed	***Volume***	***Lane***	***Density***	***GPS***	***Time stamp***	***Vehicle type***	***Vehicles counts***		
(John et al., 2016)	NFM, HMM.	Perdition.	Inductive loops.	✓				✓				MIDAS	Yes
(Kwoczek et al., 2014)	KNN.	Prediction.	floating car data.	✓				✓				Not Specified	Yes
(Julio et al., 2016)	ANN, SVM, BN.	Prediction	Chile public transport data.	✓				✓	✓			The public transportation system from Santiago, Chile	Yes
(Liang et al., 2018)	New algorithm.	Detection.	SCATS GPS data.	✓	✓		✓	✓				SCATS	No
(Jr et al., 2017)	NN, DL.	Detection.	SUMO.	✓			✓	✓			✓	SUMO	No

Reference	**Algorithm**	**Application Type**	**Data Sources**	**Used Data Attributes**								**Dataset's Name**	**Open Source**
				Speed	***Volume***	***Lane***	***Density***	***GPS***	***Time stamp***	***Vehicle type***	***Vehicles counts***		
(Polson & Sokolov, 2017)	DL.	Prediction.	Loop detector.	✓			✓		✓		✓	Illinois Departm ent of Transpor tation	Yes
(Tang et al., 2016)	EFNN.	Travel time estimation.	DRIVE NET loop detector and HERE travel time data.	✓	✓		✓		✓			HERE and DRIVEN ET	Yes
(Xuan et al., 2016)	DL.	Prediction	GPS.					✓		✓		Not Specified	No
(Tang et al., 2017)	FNN.	Prediction	Remote traffic microwa ve sensors.	✓				✓				DRIVEN ET	No

Reference	**Algorithm**	**Application Type**	**Data Sources**	**Used Data Attributes**								**Dataset's Name**	**Open Source**
				Speed	***Volume***	***Lane***	***Density***	***GPS***	***Time stamp***	***Vehicle type***	***Vehicles counts***		
(Sturari et al., 2016)	Not used	An integrated mobility system.	Camera counter, RADAR counters, induction loops, and CCTVs.					✓	✓			Not Specified	No
(Yang et al., 2015)	PSO.	Prediction.	GPS data.	✓	✓		✓	✓	✓			Not Specified	No
(Zaki & Ali-Eldin, 2018)	ANN.	Prediction	Traffic camera.	✓			✓		✓			Not Specified	No

4.4 Data Selection

Choice of variables may be critical and difficult for Traffic management applications. However, it affects the model performance and efficiency(Yuan & Tu, 2017). The variable parameters that are commonly considered for the forecasting models include, traffic flow volume, travel time and speed data. Such variables data is collected with onsite sensors using loop detectors and or laser sensors. Table 4-2 summarised the public datasets reviewed in this section with a link for the source files to view and download.

Table 4-2: Sources of the real datasets.

Reference	**Dataset's name**	**Source**	**Open source**
(John et al., 2016), (Goves et al., 2016a)	MIDAS	http://webtris.highwaysengland.co.uk/Home	Yes
(Kumar et al., 2013), **(Kumar et al., 2015)**	National highways in north India	https://journals.vgtu.lt/index.php/Transport/article/view/1708/1332	Yes
(Chang et al., 2012)	Korea Expressway Corporation	N/A	N/A
(Sokolov, 2017)	Illinois Department of Transportation	http://www.travelmidwest.com/	Yes
(Tang et al., 2016)	DRIVENET	http://www.uwdrive.net/about.html	
(Tang et al., 2016)	HERE	https://www.here.com/	No
(Lee & Min, 2017)	A map	https://ditu.amap.com/	Yes

Based on the review of used datasets the researcher chooses to use an open-source dataset to be applied in her thesis. A list of potential open datasets that are suitable for the implementation is shown in Table 4-2. Furthermore, the discussed datasets in this chapter, have enough information for the proposed model that can be useful in validation and testing processes. On site sensor loops and radar technology that were used to log the data at discrete points make the collected data a complete set of suitable datasets for the research methodology.

The chosen datasets were provided by Highways England. The data files can be found on their website and is licensed under the Open Government Licence v3.0. The data are collected using traffic sensors, mainly inductive loops for each lane across the UK. Highways England is responsible for most of motorways and major category A roads in England. Highways England has outsourced its National Traffic Information Service (NTIS). NTIS is responsible for fixed equipment at every national control centre to work well with the several subsystems of the Highways England Traffic Management Systems (HATMS). This will allow access to the Motorway Incident Detection and Automatic Signalling (MIDAS) traffic data. It also can gain alerts if there are more demands that usual. In addition, traffic data are being monitored using automatic number plate recognition (ANPR) cameras located at strategic locations on the road. Furthermore, the collection of the data is done every 5-minute. NTIS collects traffic data from various sensors and make them available in two different forms as isolated sensors data or fused sensors data (Units, 2018).

The main advantage of this data in addition to is availability under the open source licensed, is its interactive site that allows users to choose the areas(roads) that they want to check its status and view the latest traffic report for a desired dates and period of time for the status report. Due to the type, availability, and format of the data, it was decided to consider the UK traffic road networks as a case study for this research. In Section 5.5.1, a detailed description of the used datasets is presented.

4.5 Summary

The datasets play an important role in machine learning based solutions. A better model is obtained with well identified prepared data. Hence, the author focused on identifying the current datasets to be used to test the proposed framework. To achieve this task, a list of criteria was introduced that focused on the past decade vehicle and road traffic data to review the current datasets. Another step is to filter all the outcome results by ignoring any other traffic management data that is not related to transport domain but other domains such as network traffic, air traffic, railway traffic. Furthermore, the remining publications results then organised by its availability to open and nun opened source. Moreover, one of the most challenging problems in smart transport traffic management research is the insufficiency of representative and standardized data sets for anomaly detection projects. Thus, the work in hand examines the available datasets to obtain content that suits traffic detecting scenarios.

The available works that applied machine learning algorithms as a solution to traffic congestion were using either an open-source dataset from sensors, or a simulated data that used simulation tools. As a results from the reviewed datasets, this thesis will use part of the MIDAS real datasets that were collected by roadsides sensors. The datasets will be fed to the proposed framework to train and test the performance of the framework, which is presented in the upcoming chapter.

Chapter 5 A Novel Multi-level Anomaly Detection Framework

5.1 Introduction

This chapter introduces a framework that fulfils the requirements of anomaly detection in smart transport. In fact, detecting anomalies in roads is crucial for smart transport and can affect transportation in many ways (Zantalis et al., 2019). Nevertheless, there are several challenges in the field of smart transport. For instance, smart transport systems based on context-aware systems face challenges not only in processing real time data that come from a variety of sources including sensors and vehicles but also in finding relationships among collected data (Khan, 2017). Another reason for this challenge is that moving vehicles requires detection systems that can manage the complexity when collecting constantly changing real-time data (Khan, 2017). In light of this requirement, there is a need for anomaly detection techniques that consider the challenges in smart transportation systems. Therefore, this chapter aims to address the issue of anomaly detection in the domain of smart transport by describing and implementing an anomaly detection framework for detecting abnormalities in road traffic.

The chapter reports the findings of several experiments. The data preparation for these experiments involved data pre-processing, which included categorising road congestion into speed-related levels, deciding which attributes to feed into the HTM-based framework, and determining the ratio of the train-test split.

The chapter is organised as follows. Section 5.2 outlines the requirements for a smart transport anomaly detection application, and Section 5.3 details the proposed HTM-based framework. Section 5.4 sketches the methodology adopted in running the experiments, including the data pre-processing and evaluation metrics. Section 5.5 describes the proposed Multi-level anomaly detection framework experiments and also reports the findings of these sets of experiments. Section 5.6 summarises the main highlights of the chapter.

5.2 Requirements for smart transport traffic management systems

As revealed in the literature review in the previous chapters, traffic congestion management systems are forms of real time applications. Such applications need online algorithms to properly address traffic congestion. Among the various types of online algorithms, the HTM algorithm can best be implemented for traffic management systems because, unlike other online algorithms, HTM is more succinct. More specifically, HTM does not use batch data for learning and does not need pre-training. Instead, HTM consciously learns from the data stream. As a result, HTM can cope with new data patterns and adapt to rapid changes in real-time traffic data.

Algorithms are basic components of machine learning based models. To ensure effective performance for machine learning based applications, algorithms should be carefully chosen. The choice of ML algorithms depends on the nature of the application domain. Therefore, based on the review conducted in this study, HTM is the target algorithm to be trained, tested, and finally, evaluated in this thesis. In the case of the transport domain, the requirements for ML algorithms to detect and predict traffic in smart transport traffic management systems can be summarised as follows:

- Traffic management systems should integrate and learn patterns from datasets and predict future events.
- An essential requirement for traffic management systems is to be able to learn from historical data and attempt to detect abnormalities in future events.
- Traffic management systems should deal with and adapt to changes in the input.

The HTM algorithm was chosen in this study because it has proven to function well for applications in many domains as discussed in Chapter 3. Moreover, the HTM algorithm are also suitable for traffic management applications. This is because this type of algorism has features that enable it to use less memory and adapt quickly to changes in data patterns. Such features qualify the HTM algorithm to meet the essential requirements of traffic management applications. In addition, the HTM algorithm is an online algorithm that can be used in the field of smart transport particularly in traffic management applications that aim to manage traffic properly.

Along with ML algorithms that constitute a part of ML based models, other components that should also be taking into consideration. For example, a standard ML based model may include a number of steps as shown in Figure 5-1 below.

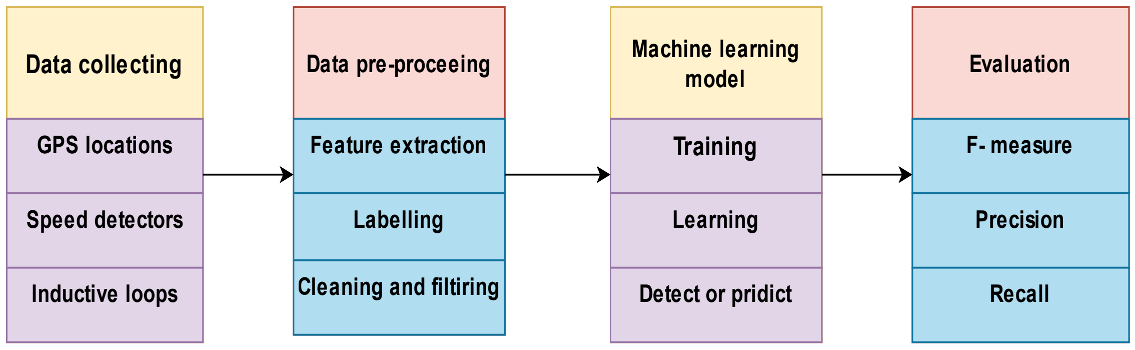


Figure 5-1: Machine learning based model.

Figure 5-1 describes a standard ML based model. As shown in the figure, the first step is to identify the datasets that fit the scenario in hand from different sources. Then, the datasets should be pre-processed to acquire the features that suit the proposal. This step includes labelling data, which is a crucial step in any ML based solution as it affects the accuracy of the trained algorithm. The next step is to train the algorithm using the pre-processed data to train the ML model. In this step, the model is trained to learn different patterns of inputs. After that, the trained model can detect abnormalities in any given data. The final step is the evaluation process, which occurs at the end of the model. In this step, different mechanisms are used to calculate the accuracy of the model. The chosen algorithm is then compared with other algorithms that can be used in the same context.

5.3 Proposed HTM

The HTM is an algorithm that can model the human brain in a very realistic way. Thus, HTM can learn and predict complex data patterns and perform complex tasks. As a self-learning algorithm, HTM does not require inference as it can continuously learn from any new input patterns. Thus, HTM algorithms do not have to have a separate inference mode (Zhang, Zhang & Chen, 2017). HTM matches the streaming inputs to fragments of sequences that have been previously learned; thus, recognised sequences are continuously predicted. As an online learning algorithm, HTM requires less training and consequently needs no pre-processing.

This thesis chapter proposes a framework for smart transport that utilizes an HTM algorithm to detect and predict traffic. The framework will be designed to learn to detect current anomalies in traffic flow and predict possible traffic anomalies in the future.

In more detail, the proposed framework will learn from the datasets, after which it will be able to identify data patterns. More specifically, the proposed framework will differentiate between events that are similar to and match with the previously learnt events and non-matching events. In this way, the framework can detect normal and abnormal traffic flows.

Before the experiments, the datasets were divided into two parts. The first part was used for training purposes whereas the second part was used for validating and testing. For the training process, 80% of the data was used, and the remaining 20% was used for testing and validating.

The experiments proposed in this study were conducted in:

1. Single HTM anomaly detection model as base-line experiments.
2. Multi HTM s anomaly detection models framework.
3. Combined anomaly detection and prediction with feedback framework proposal (see Chapter 6).

The first experiments aimed to identify the best model parameter and HTM settings for the input streams. The second and third experiments were proposed to improve the HTM performance.

5.3.1 Preparing the CLA

The initial phase in the creation of CLA models is the parameter optimization stage which is also known as ‘swarming’. The swarming applies the Particle Swarm Optimization to evaluate the performance of the model for a particular task. Within the stage, the model would be executed on both the dataset search and the parameter space, optimized for the optimal parameters in the model. The next phase, training, is conducted on the model, allowing for the testing on the segment of the new dataset and enabling the model's calculation (Lee & Rajabi, 2014). The entire procedure is mainly enabled by a framework created by Numenta, which is called the Online Prediction Framework (OPF), illustrated in Figure 5-2. The OPF can be used to essentially streamline the creation, optimization, and assessment of the model.

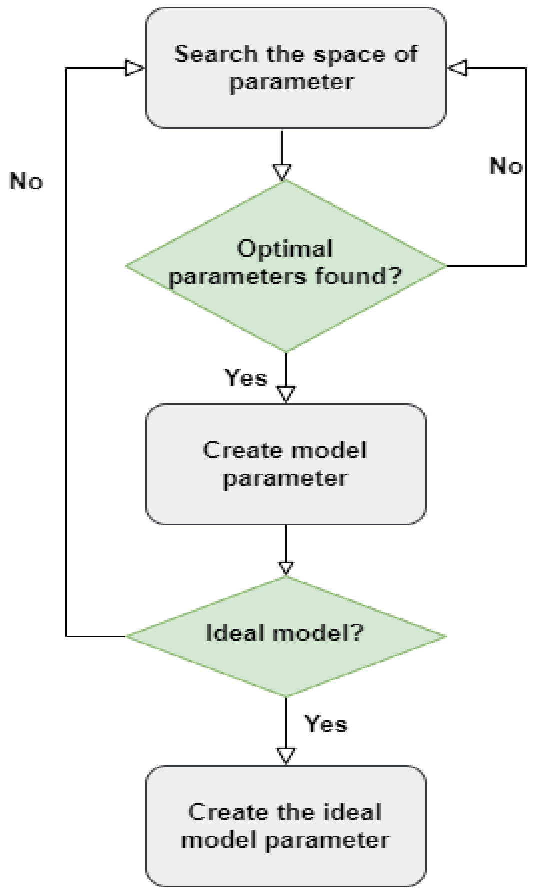

Search the space of parameter
No
No
Optimal parameters found?
Yes
Create model parameter
Ideal model?
Yes
Create the ideal model parameter

Figure 5-2: OPF Swarming flowchart.

Additionally, swarming is applied to set the required model's parameters on the dataset. As seen the figure, swarming is the process that ensures the selection of the optimal model of various options for a particular dataset. Hence, the swarming procedure can evaluate different models and generate the ideal model founded on the minimum error rate on the dataset. Also, several attributes in NuPIC need parameter values to be configured toward the model effectively. Swarming offers the ideal parameters to tune all the components like spatial poolers, encoders, classifiers, and temporal memory. These stages are critical for the creation of NuPIC models.

Besides, swarming specifies the inference form, such as temporal anomaly and encoder settings, the encoder type like a scalar, date, and category encoders, W, which is the width and number of 'On' bits', and N, which is the number of bits to represent the input which were explained in Section 3.8. The parameters of spatial pooler comprise of column count, which represents the cell column numbers within the cortical region, the maximum number of active columns within the output of the spatial pooler, and the active column numbers per inhibition region, the synapse permanence connected, which are the default linked threshold, and potential proportion, which is the percentage of the column approachable field for the potential synapses. Therefore, the synapse with the permanence value more than the linked threshold is perceived as a 'linked synapse', implying that it could essentially contributing to the cell's firing.

On the other hand, the temporal memory parameters encompass the Column Count, which is the cell columns numbers within the cortical area, and a similar number as the spatial pooler, the Cells Per Column, which represents the cells number like states assigned per column, Max Segment Per Cell that represents the maximum number of segments per cell, Max Synapse Per Segment, which is the maximum synapse number per segment, Permanence Increment, Initial Perm, which is the initial performance, Permanence Decrement, Activation Threshold, which represents the segment activation threshold, the section is perceived as active when it is above the threshold, and Min Threshold, which is the minimum number of the active synapses for the segment deemed in the search for the ideal-matching segments.

The second stage begins after the creation of the model and is perceived as ready to begin learning. This phase begins by facilitating learning, which indicates the state of training. After the phase, the model would have effectively learned the normal and standard events and is considered ready to detect the anomaly and perform prediction. Lastly, with the completion of the preparation, the learning is deactivated, and the model can begin the irregularity prediction phase. In this step, the model can detect normal and abnormal events.

5.4 Evaluation Methodologies

This section details the methodological aspects of the study. The evaluation methodology presented here follows standard practices in the discipline. It begins with an overview of the selected dataset, including a justification for the choice of dataset and a definition of the dataset parameters that are relevant to the study. Next, the data pre-processing is described. In this sub-section, a break-down of road congestion into speed-related levels is proposed; the specific attributes that are fed into the HTM-based framework are identified; and the ratio of the train-test division of the dataset is determined. Following that, the evaluation metrics to be used in the study are defined, and the overall experimental setup is explained in Section 5.4.4.

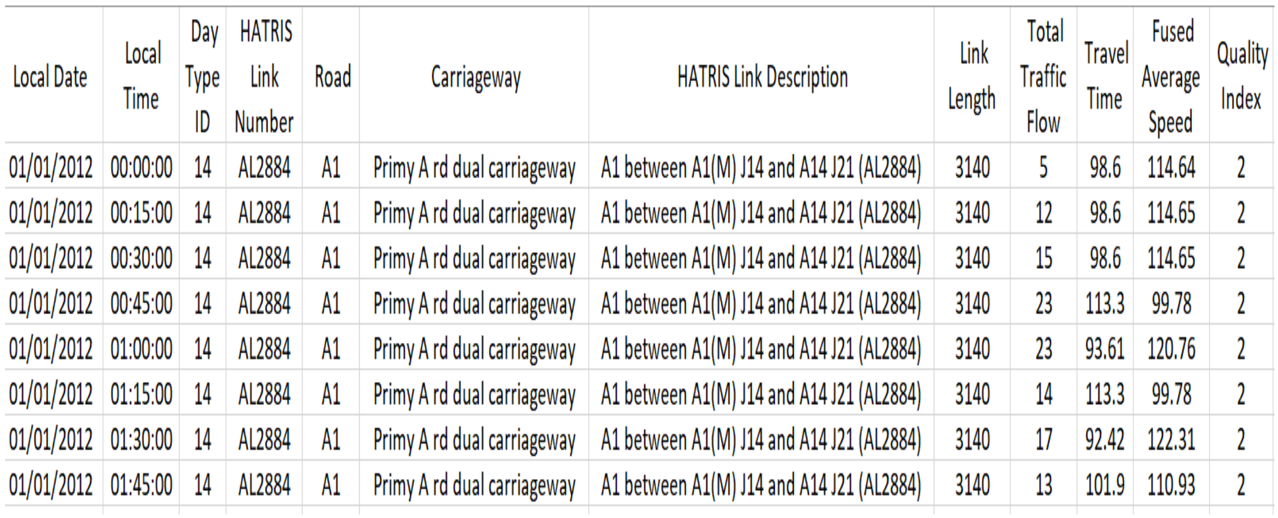
5.4.1 Dataset

The dataset used in this study comes from the MIDAS traffic dataset from the Highways England department. As was discussed in Section 4.4. The dataset exists in

CSV format. When the researcher first embarked on this research project, the most recent published dataset available was for the year 2013. However, it only covered three quarters of the year. For this reason, the 2013 dataset was dismissed. A more complete dataset that covered all twelve months of the year was deemed more suitable for the purposes of the current study. The second most recent dataset available at that time was for the year 2012, and it was selected for investigation. The portion of the 2012 dataset that was selected for this study comprises 33,890 records over a span of twelve months from January 1st to December 31st of the year 2012. There is a separate file for each month of the calendar year. Each of these monthly files contains the traffic conditions of each day in the month from 00:00:00 am to 11:15:00 pm for a total of 2880 records per month. The only exception is the file for July, which only has 2210 records, with 670 missing. The records are updated every 15 minutes during each 24 hours. Figure 5-3 shows a sample of the original CSV file source.

The file also includes data that pertains to other temporal, spatial, and dynamic parameters that are of great interest to the current study[5]. The temporal parameters are as follows: (1) Local Date, which is simply the calendar date of the event; (2) Local Time, which refers to the time of the event as a 15-minute interval in British Summer Time; (3) Day Type ID, which is a pre-specified designation of the day of the event, such as school holiday, normal working day, Christmas Day/New Year’s Day,

[5] The Highways England traffic data file also has a parameter called Link Number, which is basically a unique reference number that identifies each link. This parameter, however, is not relevant to the current study. Also in the traffic data file is a validity parameter called Quality Index. Its main purpose is to assess the quality of the data recorded. However, examining this parameter lies outside the scope of this study.



Local Date	Local Time	Day Type ID	HATRIS Link Number	Road	Carriageway	HATRIS Link Description	Link Length	Total Traffic Flow	Travel Time	Fused Average Speed	Quality Index
01/01/2012	00:00:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	5	98.6	114.64	2
01/01/2012	00:15:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	12	98.6	114.65	2
01/01/2012	00:30:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	15	98.6	114.65	2
01/01/2012	00:45:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	23	113.3	99.78	2
01/01/2012	01:00:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	23	93.61	120.76	2
01/01/2012	01:15:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	14	113.3	99.78	2
01/01/2012	01:30:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	17	92.42	122.31	2
01/01/2012	01:45:00	14	AL2884	A1	Primy A rd dual carriageway	A1 between A1(M) J14 and A14 J21 (AL2884)	3140	13	101.9	110.93	2

Figure 5-3: Sample of the CSV data file, which contains public sector information licensed under the Open Government License v3.01.

and bank holiday; and (4) Travel Time, which is given as the average time calculated in seconds that vehicles take to cross the whole length of the Link during the 15-minute time interval.

The spatial parameters in the Highways England traffic data file are as follows: (1) Road Carriageway, which is the name of the carriageway; (2) Link Description, which, as the name suggests, gives a description of each link; and (3) Link Length, which simply states the length of each link.

The dynamic parameters in the traffic data file, which are of paramount importance to the current study, include (1) Total Traffic Flow, which is the average number of vehicles that are detected on the link during the 15-minute time slot; and (2) Fused Average Speed, which is taken as the average speed in km/h of the vehicles on the Link for the 15-minute time slot.

The Highways England traffic data cover all highways in England. However, this study analyses only A1 road data. Figure 5-4 is a map of Britain depicting the A1 as it stretches from London in the southeast to Edinburgh in the northeast. The A1 is the longest road in the UK. According to a UK Government press release published on 25 March 2019, the A1 was, at the time of the press release, 410 miles (approximately 660 Km) long. More than 31,000 vehicles travel on the road every day (GOV.UK, 2019). The A1 is a dual carriageway. This study will analyse traffic data in a unidirectional flow, which proceeds along two lanes in the road.

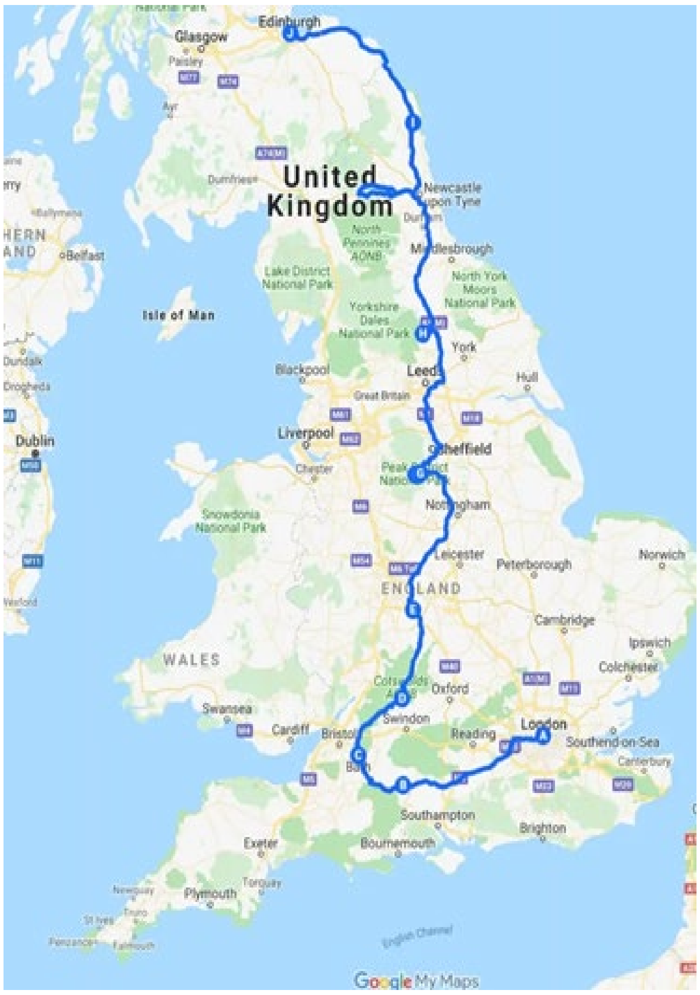


Figure 5-4: Map of Britain depicting the A1 road connecting London to Edinburgh

(Image Source: GoogleMyMap).

5.4.2 The Datasets Pre-Processing

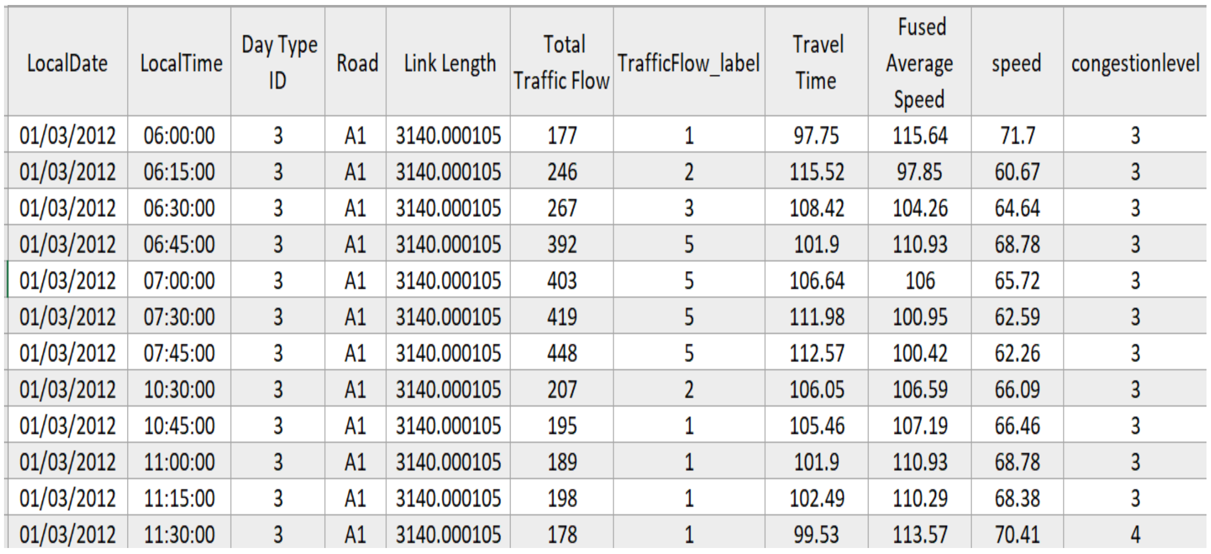
data pre-processing consists of identifying the normal flow state, the level of congestion on the road, and the specific attributes that will be fed into the HTM framework. Firstly, to calculate the normal flow value, the attribute of Fused Average Speed was converted from kilometres/hour to miles/hour. Secondly, to calculate the level of congestion, the researcher used the maximum speed limit on UK highways, which is 70 miles per hour, to generate Pseudocode 1 below. Thus, effectively, there is no congestion if the speed is between 60 and 70 miles per hour. On the other hand, a speed less than or equal to 50 miles per hour would be taken as a sign of congestion on this road. Then a scale of 1 to 5 was set up, with 1 indicating congestion and 5 indicating normal flow. The result of this calculation is shown in the pseudocode below.

Pseudocode1: Calculation of the **Congestion_Label** in the dataset, where **S** is the speed in miles/hour

```
for each row in the data do
    if( S > 80)          then  Congestion_Label = ”5”
    if(80 <= S > 70)  then Congestion_Label = ” 4”
    if (70 >= S > 60)   then Congestion_Label = ” 3”
    if (60 >= S > 50)   then Congestion_Label = ” 2”
    if (50 >= S)          then Congestion_Label = ”1”
  end if
end for
```

Thirdly, care was taken in selecting the attributes that would be fed into the HTM framework to obtain the results. This step is vital as the output results and their interpretation greatly rely on the selected data. The selection criteria are in total alignment with the aims and objectives of the project and are compatible with the nature of HTM algorithms.

Given that the HTM algorithm works well with time dynamics as it is an online algorithm, the following attributes were selected based on the commonly used datasets attributed from datasets reviewed in Section 4.3: (1) Local Time, (2) Traffic Flow, (3) Speed, (4) Congestion Level, and (5) Traffic Flow label. Figure 5-5 shows a sample of the datasets after pre-processing was completed.



LocalDate	LocalTime	Day Type ID	Road	Link Length	Total Traffic Flow	TrafficFlow_label	Travel Time	Fused Average Speed	speed	congestionlevel
01/03/2012	06:00:00	3	A1	3140.000105	177	1	97.75	115.64	71.7	3
01/03/2012	06:15:00	3	A1	3140.000105	246	2	115.52	97.85	60.67	3
01/03/2012	06:30:00	3	A1	3140.000105	267	3	108.42	104.26	64.64	3
01/03/2012	06:45:00	3	A1	3140.000105	392	5	101.9	110.93	68.78	3
01/03/2012	07:00:00	3	A1	3140.000105	403	5	106.64	106	65.72	3
01/03/2012	07:30:00	3	A1	3140.000105	419	5	111.98	100.95	62.59	3
01/03/2012	07:45:00	3	A1	3140.000105	448	5	112.57	100.42	62.26	3
01/03/2012	10:30:00	3	A1	3140.000105	207	2	106.05	106.59	66.09	3
01/03/2012	10:45:00	3	A1	3140.000105	195	1	105.46	107.19	66.46	3
01/03/2012	11:00:00	3	A1	3140.000105	189	1	101.9	110.93	68.78	3
01/03/2012	11:15:00	3	A1	3140.000105	198	1	102.49	110.29	68.38	3
01/03/2012	11:30:00	3	A1	3140.000105	178	1	99.53	113.57	70.41	4

Figure 5-5: Sample of the pre-processed CSV data file showing the congestion level and traffic flow label attributes.

A final step in this stage is to divide the pre-processed data files into two sets: training data and testing data. The training dataset constitutes around 80% of the entire dataset. The remaining 20% of the dataset will be allocated for validation and testing.

5.4.3 Evaluation Metrics

Many different metrics are used to evaluate the performance of ML algorithms. These include Precision, Recall, F-score (F-measure), and Accuracy (Ozbayoglu et al., 2016)(Powers, 2020). Of these metrics, Precision and Recall are the most commonly used evaluation metrics for time series applications (Tatbul et al., 2018). In ML classification tasks, prediction performance yields four possible logical outcomes, including (1) true positives, where a case is correctly predicted to belong to its class; (2) true negatives, where a case that does not belong to a class is correctly predicted not to belong to that class; (3) false positives, which are obtained when a case which does not belong to a class is nevertheless wrongly predicted to belong to that class; and (4) false negatives, which occur when a case that belongs to a class is wrongly predicted not to belong to that class.

Given the confusion matrix terms above, Precision is often defined as the proportion of true positives to all detected positive cases, both true and false. Conversely, Recall is often defined as the proportion of detected true positives to all cases that should be detected (true positives and false negatives). As such, Precision can be seen as a positive predictive, while Recall can be seen as a sensitivity measure. As far as anomaly detection performance evaluation is concerned, Recall is the percentage of all detected anomalies which are identified as real anomalies which is represented as true positives (TP) and false negatives cases (FN) , whereas Precision is the percentage of all real anomalies that have been identified as anomaly successfully which is called TP (Tatbul et al., 2018). For example, consider a number of points anomalies data illustrated Figure 5-6, where the blue points are FN, purple are TP, and reds are FP that were explained earlier in this section. In the case of Precision, is represents the circle that contains the red and purple points which means that all real anomalies (TP) or not anomalies (FP). While in the case of Recall, the points that represents all TP and FN points. Furthermore, these metrics are usually calculated together; they are treated as complementary in the sense that Precision seems to cover aspects of quality, while Recall covers aspects of quantity in the evaluation task in question.

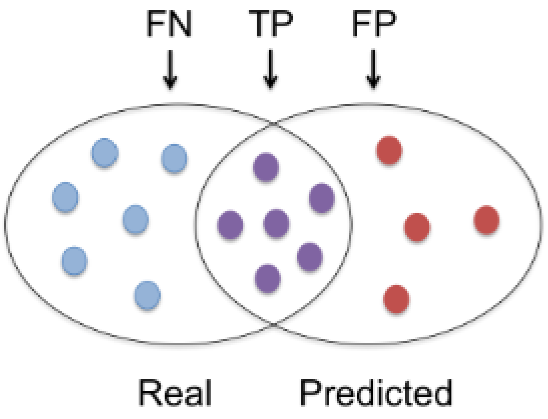


Figure 5-6: Precision and Recall concept adapted from (Tatbul et al., 2018).

In addition, Precision and Recall are popular measures in information retrieval tasks (Lee et al., 2018). As a matter of fact, these metrics were first devised and popularised in the field of information science (Baeza-Yates & Ribeiro-Neto, 2011; Tatbul et al., 2018). Precision and Recall have long been used to evaluate many ML-based projects. The two metrics have steadily gained credibility. A notable example in this regard is the comprehensive study conducted by Rajeswari et al(2018). The authors set out to evaluate the performance of different supervised and unsupervised algorithms using different datasets. Their datasets include traffic incidents, education, diabetes data, and stock market data. Rajeswari et al. (2018) concluded that Recall is needed for better outlier detection, whereas precision is needed for the evaluation of the classification accuracy of the algorithm under investigation. However, the researchers also call for the integration of supervised and unsupervised techniques in a hybrid outlier detection approach.

Finally, a balanced combination of the two metrics of Precision and Recall is formally expressed as the F-measure, which is basically the equally weighted average of Precision and Recall.

Another important performance evaluation metric is Accuracy. It is basically the ratio of correct predictions to the total cases in the dataset under evaluation. As far as road traffic performance evaluation metrics are concerned, Precision, Recall, the F-measure, and Accuracy are widely used (Liu et al., 2016; Silva et al., 2020). These performance evaluation metrics are obtained using the obtained using the equations in 5.4-1, 5.4-2, 5.4-3, and 5.4-4.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \qquad 5.4\text{-}1$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad 5.4\text{-}2$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad 5.4\text{-}3$$

$$\text{F} - \text{measure} = 2\left(\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right) \qquad 5.4\text{-}4$$

where TP is the True Positives, TN is the True Negatives, FN is the False Negatives, and FP is the False Positives.

5.5 The Proposed Multi-Level HTM Based Framework For Anomaly Detection

One of the main contributions to knowledge in this study is the proposal of a framework that provides better anomaly detection and prediction for traffic. This framework is based on the standard CLA presented above. The novelty of this framework is that it is based on a combination of different anomalies to be used as a predictive model to detect anomalies and then, based on that detection, it predicts what will happen next. The proposed model is shown in Figure 5-7.

The steps to design and implement the framework in Figure 5-7 are summarised as follows. The two levels of anomaly detection named L1 and L2 for simplicity. In the first level, the inputs are the speed, congestion level, and time, which then fed into the L1 model. In this step, Input encoded by an encoder to form a Sparse Distributed Representations (SDRs). SP then identifies similar sequences and learns them, which involve initialising the HTM region, compute the overlap to decide whether a column is active or not, and inhibition to control how many columns will be returned as active. After that, TM will get the new SDRs from the SP. This step involves learns sequences of SDRs that comes from SP and performing prediction from learned sequences. Once the SP and TM are initialised, the model will be ready to learns new sequences and start to identify anomalies from the input data. Similar steps will be conducted by the second level L2 but with the input traffic flow, TFL which represents the traffic flow levels, and time stamp. Finally, the outcomes of L1 and L2 are concatenated to a new anomaly detection model called L3. This concatenated level then produces the anomaly scores which they can be used to calculate the performance of the framework.

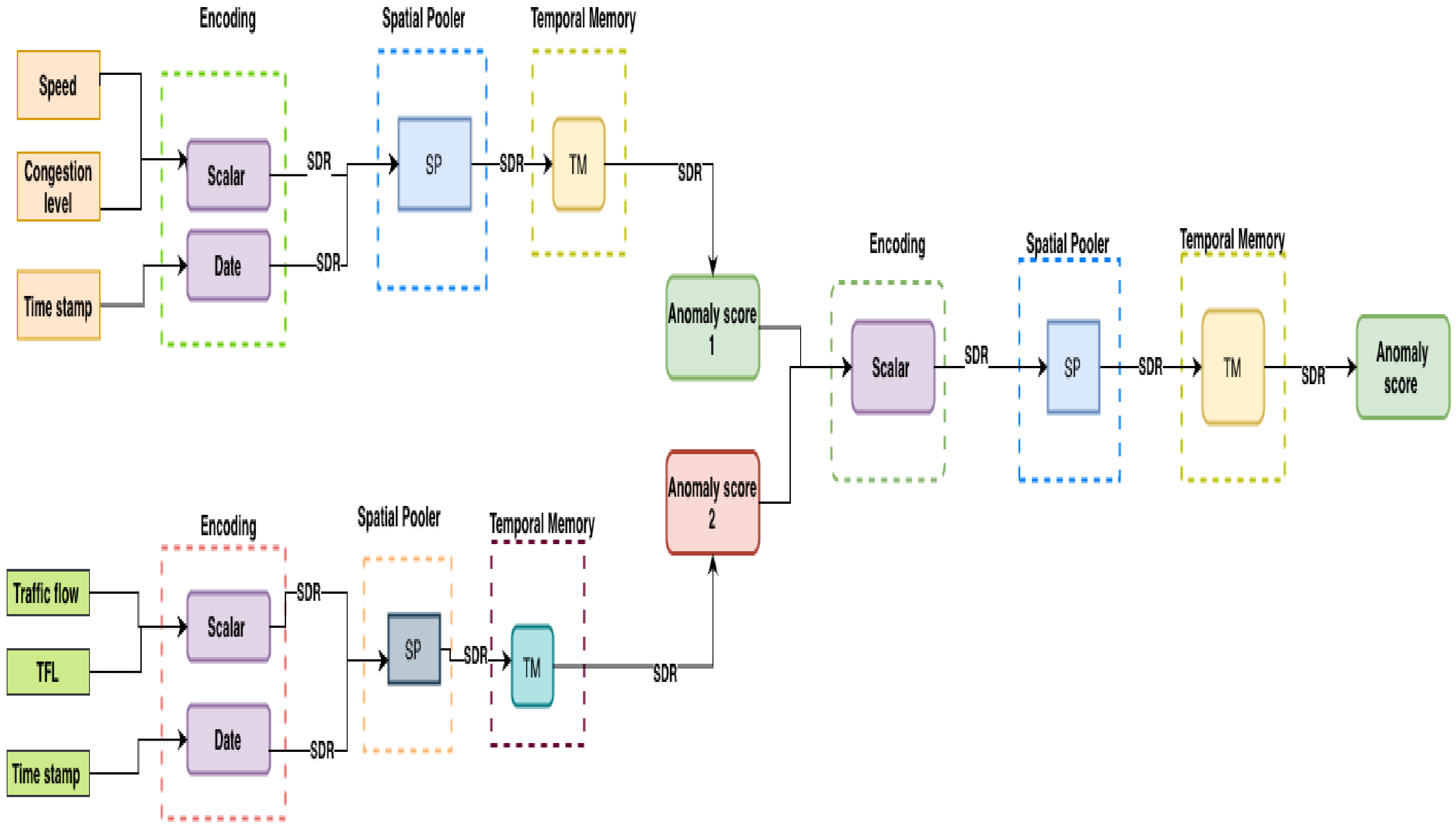


Figure 5-7: The proposed multi-level anomaly detection framework.

The designed Framework will benefit from the HTM hierarchy feature to learn patterns quickly, and then predict the next future step based on its previous knowledge. In this project, two models to detect anomalies in the input was used. After that, the combination of the outcome anomaly scores were fed to a new anomaly model to check for any undetected anomalies and then based on the new model the possible congestion in the future can be predicted.

The designed Framework was implemented under the Numenta Platform for Intelligent Computing (NuPIC) framework. NuPIC is an open source from Numenta that implements the HTM codebase that was structured to run algorithms in Python (Numenta, 2018).

The algorithm was trained using 80% of the data that represent normal traffic when there is no congestion. Then, the rest of the data, which represent normal and abnormal traffic, was fed to the trained algorithm to identify any anomalies and then provide the anomaly score. Results from this stage were subsequently evaluated.

5.5.1 Context

The rationale behind deploying Hierarchical Temporal Memory (HTM) rests on the vast potential and capabilities of HTM that can be leveraged to find solutions for traffic detection and management issues that remain unresolved. There is solid ground say that perhaps the one characteristic of HTM that particularly makes it a good option for anomaly detection is its huge capacity to continuously integrate both remotely and recently learned contextual information for predicting and detecting unusual patterns in both univariate and multivariate datasets (Wu et al., 2018).

As detailed in Chapter 3 of this thesis, it is important not to lose sight of the fact that HTM has its origins in neuroscience and biology and that HTM emulates the cognitive and sensing functions and performance of neocortex learning in the human brain (Agrawal & Franklin, 2014; Hawkins et al., 2010; Hawkins & George, 2006; Hawkins & Blakeslee, 2004). Furthermore, time, feedback, and the physical hierarchical structure of the brain are among the important aspects of intelligence which HTM provides. This could bring in the possibility of real-life scenarios into a reality in traffic detection applications. In fact, this capacity of inference making that is informed by temporal patterns is a major justification for deploying HTM in this study. Other equally important features of HTM that was found in Section 3.7.2 is its tolerance to noise and adaptability, which can improve the performance even though there are noisy or missing data streams due to the hierarchal structure (Wu et al., 2022). These features add to the reliability, efficiency, and robustness of the algorithm. In a nutshell, as this baseline experiment demonstrates, HTM is a realistically promising option to adopt for the optimisation of traffic detection and management.

Similarly, CLA, which also functions on a data structure, provides spatial and temporal pattern recognition. This characteristic makes it a strategically good option to deploy for traffic detection and management (Townsend & Gong, 2018).

In traffic management applications, congestion is increasingly becoming a commonly occurring problem for highways and urban ways, alike (Tisljaric et al., 2020;Yujun et al., 2019). An ultimate goal is to have a seamless flow during rush hours. Early detection of abnormal flow can greatly help reduce or pre-empt many congestion events. With its promising capabilities as briefly reiterated above and explained in detail in Chapter 3, HTM-based traffic management system guarantees early detection of abnormal traffic flow.

One major attractive feature of multi-level structure, according to Hawkins et al. (2011), is the rationalisation of both memory usage and training time. This type of structure is highly efficient in that the information that is extracted at any level of the hierarchy is reused when merged at the upper levels. The thesis implements a multi-level hierarchy. The idea is to get a clear version of the traffic condition by letting the information arises though the hierarchy. The outcomes of the lower level will be used to the upper level in the regions from the next heretical level.

In the experiments reported in the following sections, the researcher proposed a framework that integrates different anomaly detection models. This can help to guarantee a seamless and smooth flow. The proposed workflow of the framework is charted in Figure 5-7. The experiments proceed along three parts: (1) a two-level anomaly detection with multi regions; (2) concatenated anomalies of the multi regions into one region; and (3) encoder parameter settings optimisation procedure was carried out. Evaluation is primarily based on Recall, Precision, Accuracy, and the F-measure metrics, as presented in section 5.4.

The results of the baseline experiments reported in Appendix A are encouraging and strongly suggest that HTM has real potential to improve the performance of traffic systems. However, one caveat concerning the time factor needs to be addressed adequately before any overall conclusive evaluation of the algorithms is made. It should be remembered that a major requirement of any traffic system is to reduce waiting time and to ease the congestion as quickly as possible. Therefore, the dataset variables are expanded, and the experiment settings are adjusted to examine the productivities of the algorithm in detecting and predicting congestion. In this context, the datasets are including the peak hours as time is a vital criteria for detecting abnormalities in traffic to guarantee a seamless flow that allows less time to spend on roads. For the first level of the framework, the experiments are conducted into 3 parts: (1) preparing the datasets to include peak time period in the training and testing sets, (2) prepare the CLA model to conduct experiments and to generate the code to run the model, (3) encoder parameter optimisation is done to find the best parameter for the framework, see Figure 5-7.

5.5.2 Data preparation

Unlike the datasets in the baseline experiments, the datasets here have been aggregated and prepared to include time of the day, as well as peak and non-peak hours. Firstly, the datasets were filtered and arranged according to peak and non-peak hours. Then the datasets were categorised and labelled according to the congestion level and total traffic. Congestion level is discretised into five categories according to speed figures within morning and evening peak traffic times. These peaks occur around 8:00:00 am to 10:00:00 am and 17:00:00 pm to 19:00:00 pm. The speed-related congestion categories are (1) no congestion, (2) medium flow, (3) high congestion, (4) very high congestion, and (5) severe congestion. On the scale, 5, which represents severe congestion, is assigned when speed is less than or equals 55m/h. A speed greater than 55m/h but less than or equals 59 m/h is labelled very high congestion and is given number 4. A speed range between 60 m/h and 65 m/h belongs to the category of high congestion and is assigned number 3. Medium flow covers a speed range of 65 m/h to 69 m/h and is given number 2. 1 on the scale represents no traffic. This is when speed is greater than or equals 70 m/h.

Traffic flow labelling and implementation proceeds along similar lines, but, this time, on a four-point scale. The datasets are arranged into four categories based on traffic flow figures. These are (1) no traffic, (2) medium flow, (3) high traffic, and (4) heavy traffic. On the scale, 4, which represents heavy traffic, is assigned when the traffic involves 350 or more cars. A traffic event involving fewer than 350 cars, but more than 250 cars is labelled high and is given number 3. A range between 250 and 201 cars belongs to the category of medium traffic and is assigned number 2. 1 on the scale represents no traffic. This is when traffic involves 200 or fewer cars.

The dataset treated in this way stretches over a period of 6 months. These records are then merged together from the original files and stored into a single CSV file. There are a total of 8050 rows to be used for the training (80%) and testing (20%) sets.

5.5.3 Experiment settings

In this stage the input rows for the model are identified. These are Speed, Traffic Flow, Congestion Label, Traffic Flow Label, and Time of the Day. The experiment was conducted in two parts, L1 and L2. In part L1, Speed, Congestion Label, and Time of the Day were used as an input. In Part L2, however, Traffic Flow, Traffic Flow Label, and Time of the Day were used. The purpose of this two-part setup is to test the suitability of the model for different kinds of input.

The CLA model is then created by the swarm and the training dataset was fed into the model. The input datasets are first encoded. Subsequently, the spatial pooling (SP) in the region receives the input that were encoded from the encoders. The SP starts to learn the features of the input, then provides an active column as the output set. Next, the temporal memory (TM) learns, predicts, and feeds the indices of the active cells.

As a secondary contribution proposed in this thesis to identify the best SP and TM parameter settings, the experiments were conducted in two parts. The first part was executed by implementing the HTM model using the NuPIC parameter settings. The second part tested various parameter settings for the model. Finally, active cells from TM were moved to the classifier. The final stage is to disable the learning and test the model using part of the dataset.

As a secondary contribution to knowledge, experiments are conducted to identify the best encoder settings for the datasets, which is the reason for conducting these experiments in this context. According to (Purdy, 2016a) the encoder is responsible for determining which output bits should be ones, and which should be zeros for a given input value to capture the important semantic. From the baseline experiments in Appendix A, the best performance was achieved when the dataset is encoding using the scalar encoder.

To the best knowledge of the author, there is almost no published work related to the parameter optimisation for the encoder, special pooler, and temporal memory in this domain except the experiments done by Bastaki (2019) who tested a number of different parameter settings that led to improve the performance of the work that was proposed. Therefore, a new experiment on the encoder parameters is conducted.

5.5.4 Encoder parameter settings and results

In this experiment, several values for the encoder were tested to identify the best setting performance before moving to the next step in the experimental part. The important of this experiment is to get the most benefit of the encoder that helps to transform the input streams into SDRs. According to (Purdy, 2016a), the encoder parameter are depending on the nature of the datasets and the application domain. In NuPIC encoders as explained before in Chapter 3, the scalar encoder is encoding a numeric value linearly into a sparse array of bits using a contiguous block of '1's. The location of this contiguous block varies continuously as the input spans its entire range; with similar input patterns characterized by common semantics assigned overlapping representations. Scalar encoders use a

number of parameters to determine the encoding for a given input:

- **Minval**: minimum value of an input.
- **Maxval**: maximum value of an input.
- **W**: width of a contiguous block of active bits.
- **N**: number of encoder output bits.

For the experimentations, the Minval and Maxval must be fixed to match the highways valuable speed limits. W and N are tested several times to obtain the performance that results in the highest accuracy. The suggested rule for setting the value for N and W is that N>= W. Another remark about W is that is must be an odd number to overcome any centring issues (Hawkins et al., 2017) to guarantee that all data are encoded.

After satisfying these conditions the experiments proceed by examining several values for N, then the performance of the values is calculated. Results are then compared to identify the best performance.

Figure 5-8 and Figure 5-9 illustrate the outcome of the performances for the values that tested for N values for the encoder. Figure 5-8 also shows that the greater the number of (N), the better the performance. The best performance was reached when the N value was set to 200 as its reached above 65% compared to the other N setting values. However, as seen in Figure 5-9, there is a steady increment in the accuracy performance for the framework. It also can be seen from the figure that the best performance was reached when (N) was equals to 200 with more that 49% accuracy performance.

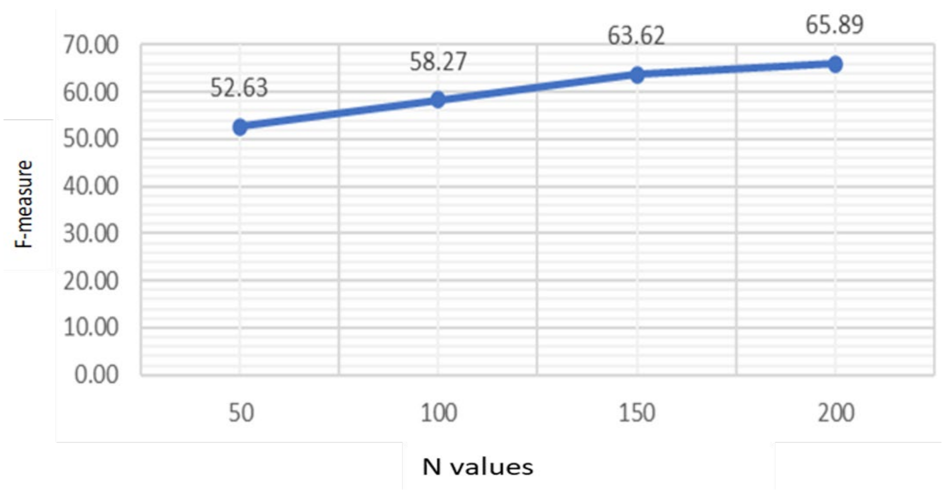


Figure 5-8: F-measure performance for N parameter.

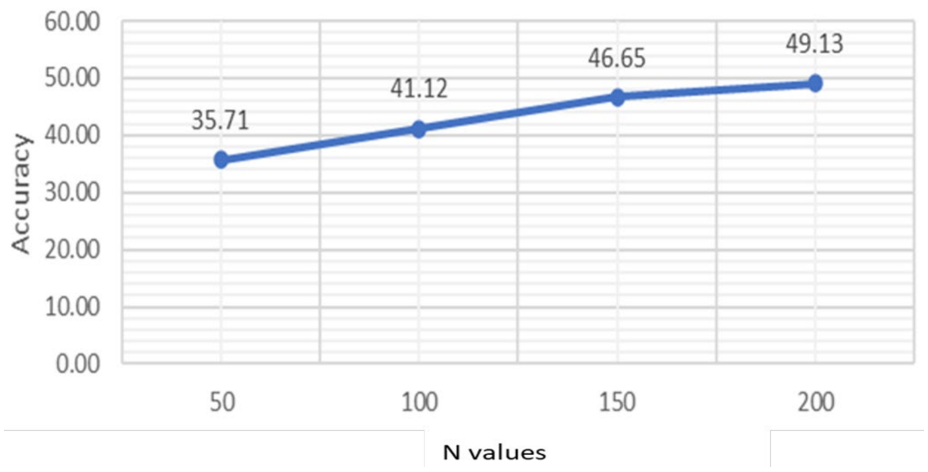


Figure 5-9: Accuracy performance for N parameter.

Then the next step is to test the W values using the same Maxval, Minval, and N sets to be 200, considering that the values should be odd numbers that are smaller than N values, as discussed above. The best performance between those values is then observed.

Figure 5-10 shows that the best performance was gained when the value of W was set to 51, while Figure 5-11 shows the F-measure at the same point. For both figures, the accuracy and F-measure started to decrease by less than 0.75%, then the performance maintained a steady increment until it reached the best performance when W was set to be 51. The accuracy was improved by almost 3% compared to the original W value, which was 21; while the F-measure was improved by 2% compared to the original W value. These improvements to the performance are beneficial to the traffic congestion as it is crucial to detect anomalies as early as possible. In this case, the data are encoded in a way that allows the framework to perform well.

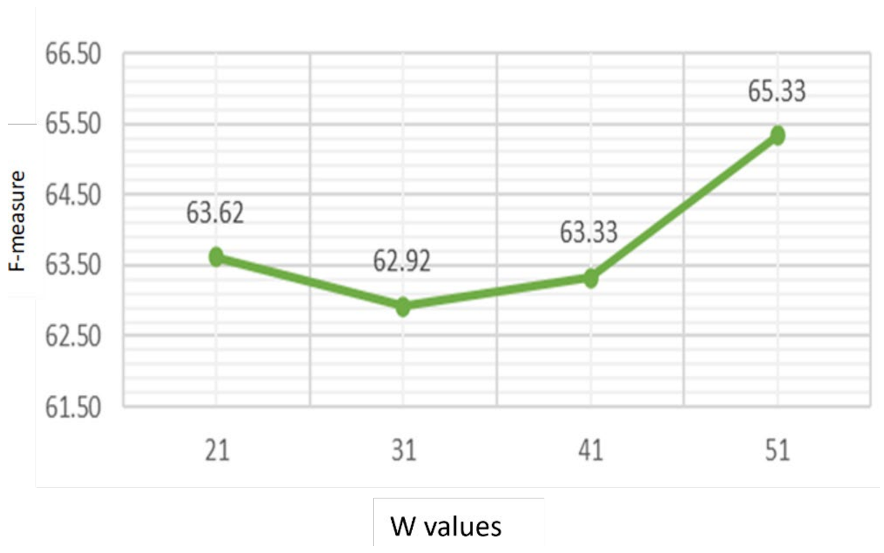


Figure 5-10: F-measure performance for W parameter.

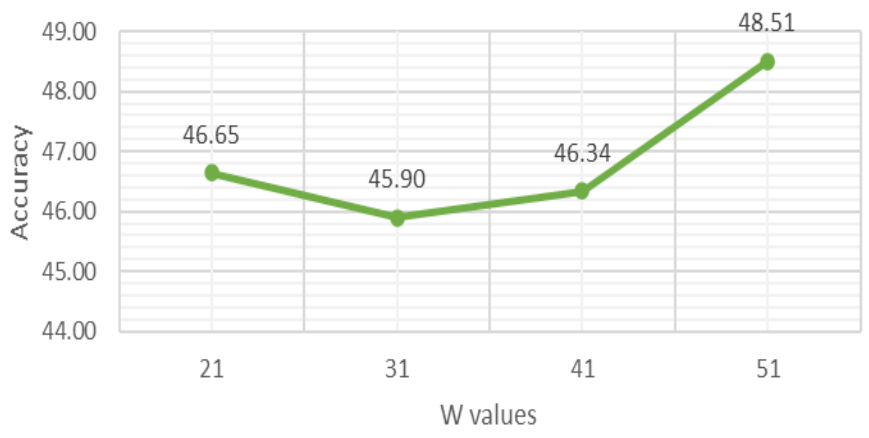


Figure 5-11: Accuracy performance for W parameter.

5.5.5 Results

The results explained in this section represent the final experiments conducted to evaluate the performance of the Multi-Level Anomaly Detection Framework after conducting the anomaly experimentations for each level, and then they are concatenated into the final level. In this section, two performances named AD_A and AD_B are compared. AD_A represents the framework performance using the best encoder parameter settings listed in Table 5-1, while AD_B represents the framework performance with the original encoder settings.

Table 5-1: Encoder Parameters settings used in experiments.

Parameter Name	**Used Value**
W	51
N	200

The resulting scores of the framework model are between 0 and 1, with 0 representing normal values. Any non-zero score is an abnormal value, which is represented by an anomaly score field. These results were granted after several experimentations to optimise the encoder parameter. These results are graphed inFigure 5-12Figure 5-14Figure 5-13.

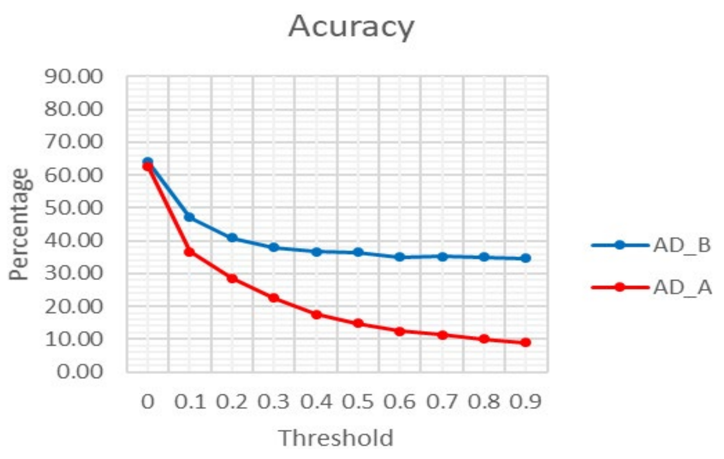


Figure 5-12: Accuracy comparison for the framework.

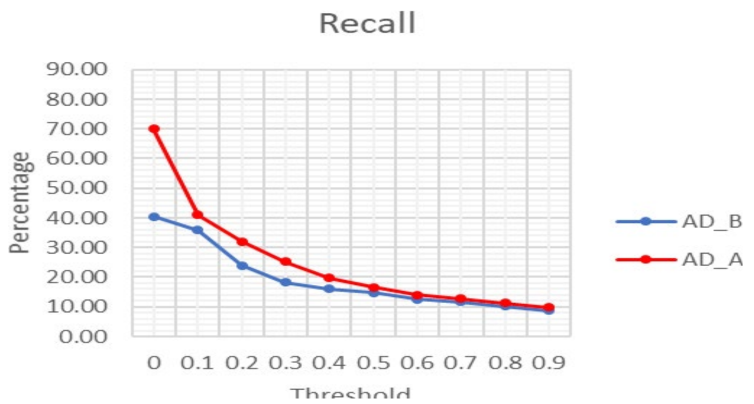


Figure 5-13: Recall comparison for the framework.

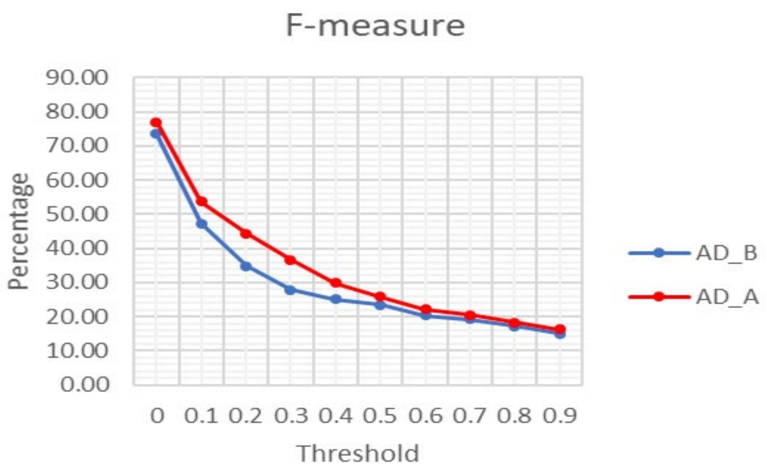


Figure 5-14: F-measure comparison for the framework.

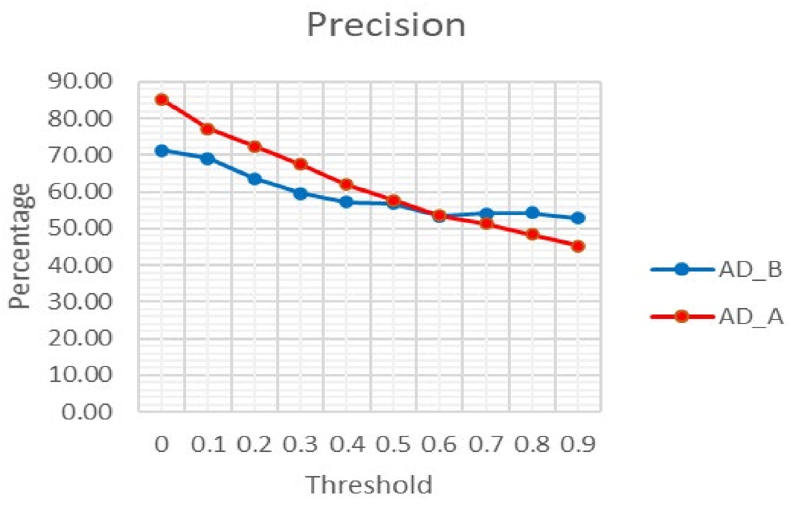


Figure 5-15: Precision comparison for the framework.

After identifying anomaly scores, the results were evaluated using the RapidMiner framework to calculate the performance evaluation metrics of Recall, Precision, Accuracy, and the F-measure that were explained in Section 5.4.3. This was done by comparing the resulting anomaly scores with a threshold that varies between 0.1 and 0.9 to determine the best performing model. The framework show improvement in anomaly detection compared to the framework with original encoder parameter settings. As illustrated in the figures, the framework AD_A scores high accuracy rates compared to AD_B.

5.6 Summary

In this chapter, a proposal has been made for a Multi-Level anomaly detection framework that provides better anomaly detection for traffic based on an HTM algorithm. The HTM hierarchical structure helps ineffectiveness, in which the memory utilization and training period is essentially reduced because, at every hierarchy level, the learned pattern is reutilized.

First, the chapter explained the requirements for smart transport traffic management systems in Section 5.2. In addition, this chapter presented the proposed HTM experiments in section 5.3. Two HTM models were used to test the algorithm's capabilities to detect anomalies in the dataset. The experiments were conducted to investigate the contribution of the scalar vs. coordinate encoders to the overall performance of the HTM-based algorithm on the performance evaluation metrics adopted in the study. The results showed that the model with the scalar encoder performed better in detecting anomalies in the dataset than the model with the coordinate encoder. Therefore, the decision was made to use the scalar encoder in the subsequent experiments that tested the proposed HTM-based anomaly detection multi-level framework.

A pre-processing stage covering data preparation and labelling was presented. The train-test division was 80% to 20%. The performance metrics used for evaluation were Precision, Recall, Accuracy, and the F-measure.

In HTM, as briefly discussed in Section 5.5.1, time perception is essential in learning, inference, and prediction since nothing is deduced from the sensory perceptions without the application of its history in time. Thus, HTM is appropriate for ideal learning from a stream of data in training. Additionally, the advantages of utilizing the projected framework.

The structure of the proposed experiments was introduced in Section 5.5 of this chapter. It was implemented and evaluated in compliance with the research questions that this study aims to investigate. Different types of experiments were conducted using the selected and pre-processed dataset from Highways England's MIDAS traffic data pools. Traffic data collected for A1 road, Britain's longest road, over a 12-month period were investigated. The temporal, spatial, and dynamic parameters of the selected data were considered. The three most important parameters of these were Total Traffic Flow, Fused Average Speed, and Time Stamp. They were generally used to determine congestion levels on the road.

After that, the encoder components were tested to ensure that the encoded datasets were encoded well, so the HTM model could detect any noise in the data streams. The results of these tests were evaluated at the end of Section 5.5.5.In the next chapter, the multi-level anomaly detection framework will be challenged to improve its performance by adding a productive model that includes the feedback from the previous anomaly to the framework. Moreover, details of the implementation and testing of an improved framework with prediction are discussed in the next chapter.

Chapter 6 The Multi-level Anomaly Detection with Prediction Feedback Framework

6.1 Introduction

The purpose of this chapter is two-fold. Firstly, to present the implementation configurations and testing results of the proposed multi-level anomaly detection with prediction feedback framework. The overall experiment involved testing a host of variables to identify the model with the best performance. Secondly, to evaluate the performance of the proposed multi-level anomaly detection with prediction feedback framework using the widely recognised performance evaluation metrics of Precision, Recall, Accuracy, and the F-measure. There is also a comparative side to the evaluation presented in this chapter. The anomaly detection performance of the proposed multi-level anomaly detection with prediction feedback framework will be compared with the performance of three well-known conventional methods. These are: the K-Nearest Neighbour Global Anomaly Score (KNN-GAS) (Asharf et al., 2020), the Independent Component Analysis-Local Outlier Probability (ICA-LoOP) (Tharwat, 2018; Krithigarani & Karthik, 2014), and the Singular Value Decomposition Influence Outlier (SVD-IO) (Khoshrou & Pauwels, 2019; Kumar & Nasser, 2012). These algorithms in particular were selected for comparison because they are the commonly used anomaly detection techniques in many domains, especially for detecting outliers in data (Chandola et al., 2009; Rajeswari et al., 2018).

In terms of the overall experimental setup, the platforms used for anomaly detection implementation and evaluation included the modern Numenta Platform for Intelligent Computing (NuPIC) version1.0.5 with its NuPIC Bindings version1.0.0, and Python version 2.7.3. Additionally, the RapidMiner Studio, version 9.3, was used to implement the conventional comparison models. This implementation is the most developed, and several commercial products have been built to run on top of the code base. As of this writing, NuPIC is considered the state of the art of HTM implementations. It is under active development by researchers and members of the community. It is also free to researcher to use.

To examine the contributions of the temporal memory and spatial pooler parameters to the performance of the proposed multi-level anomaly detection with prediction feedback framework, several testing experiments were performed to identify the best temporal memory and spatial pooler parameter settings. The experiments then proceed along three lines: (1) a multi-level anomaly detection with SP and TM parameter settings; (2) a prediction model that provides feedback to the concatenated anomalies; (3) the evaluation of the performance in these experiments based on Recall, Precision, Accuracy, and the F-measure metrics. The anomaly detection performance of the proposed framework is then compared with that of the three conventional models mentioned above.

The chapter is organised as follows. Section 6.2 details the specifics of the experiments setup, Section 6.3 details the SP and TM parameter optimisation experiments, and Section 6.4 describes the proposed framework, its experimental setup and the results.

The section also reports the findings of these sets of experiments. Section 6.5 presents an evaluation of the performance of the algorithms under investigation. The section also provides comparison figures for the performance evaluation metrics for the algorithms in this thesis. Section 6.6 discusses the experimental findings and reflects on their implications for smart transport. Section 6.7 summarises the highlights of this chapter.

6.2 Experiment Setup

In this part of the chapter, the design of the experimental work is explained. The experiments are run to assess the performance of the proposed framework, as well as the performance of the conventional machine learning anomaly detection models.

In Chapter 5, the multi-level framework was explained. In this chapter, the improved framework with prediction with feedback were tested according to the design in Figure 6-1. This figure shows the steps that are followed to test and evaluate the performance of the proposed work against several conventional ML methods, which are summarised as follows.

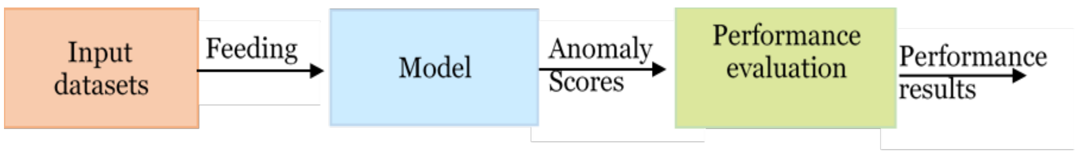


Figure 6-1: Overall experimental design.

First, data were pre-processed and fed into the model, as detailed in Section 5.5.2. The dataset was divided into two sets: one is for running the baseline experiment, while the other is set aside for testing and validating the proposed framework and the conventional anomaly detection algorithms.

The ’Model’ box in the figure represents all models used in this project, including the proposed framework and conventional methods. The models were trained, and then the anomaly scores that represent the detection of abnormal cases are obtained. These anomaly scores are then used to calculate the performance evaluation metrics for the models. As spelled out in Section 5.4.3 in Chapter 5, Precision, Recall, Accuracy, and the F-measure are used to evaluate the performance of the models. The platforms used for the implementation and evaluation of the models are (1) the NuPIC[6] platform, version1.0.5, with NuPIC Bindings, version1.0.0, and (2) Python version 2.7.3. In addition, RapidMiner[7] Studio, version 9.3, is used to implement the conventional anomaly detection ML methods.

6.3 SP and TM Parameters Optimisation for the Framework

In this section, the optimal parameter settings for the SP and TM algorithms are identified so that the performance of the proposed framework can be accurately measured.

In the experiments discussed in Chapter 5, there were no differences between the spatial pooler and temporal pooler parameter settings, as the HTM model in question was built using the recommended NuPIC settings. However, one of the contributions to knowledge proposed in this thesis was to identify the best parameter settings; hence, additional experiments to fulfil this contribution.

The rationale behind conducting these experiments is that according to (Hawkins et al., 2017) the default special pooler and temporal memory settings published by Numenta

[6] http://nupic.docs.numenta.org/stable
[7] https://rapidminer.com/products/studio/

are non-domain specific; hence, Numenta encouraged researchers to alter these settings to their specific domain requirements.

In Section 5.5, there are two levels of anomaly detection regions have allocated spatial pooler and temporal memory algorithms which are used to learn spatial and temporal patterns of the input space. Therefore, the settings of the first level were optimised first for the spatial pooler and temporal memory then the same was done for the second level. In Figure 6-2, the parameter optimisation workflow and experimental test plan show the three phases of identifying the best parameter settings. Several experiments were performed in each phase, and the model was run with different settings to compare the outcomes.

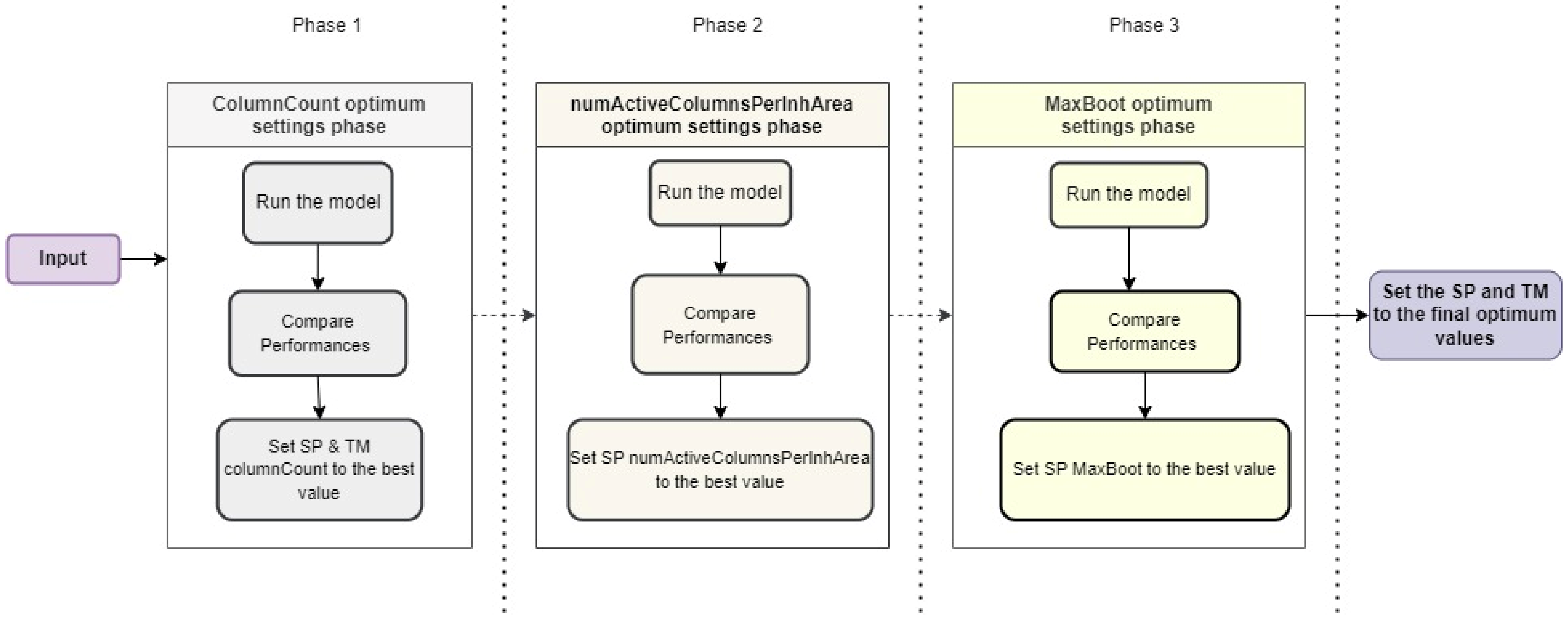

Phase 1
Phase 2
Phase 3
Input
ColumnCount optimum settings phase
Run the model
Compare Performances
Set SP & TM columnCount to the best value
numActiveColumnsPerInhArea optimum settings phase
Run the model
Compare Performances
Set SP numActiveColumnsPerInhArea to the best value
MaxBoot optimum settings phase
Run the model
Compare Performances
Set SP MaxBoot to the best value
Set the SP and TM to the final optimum values

Figure 6-2: Parameter optimisations and experimental workflow.

In the first phase, the column count parameter (ColumnCount) in the spatial pooler and temporal memory algorithms is changed to find its best performance. The column count parameter is defining the total number of columns in the spatial pooler and temporal memory as mentioned in Chapter 3. According to (Hawkins et al., 2017) the value of the column count is dependent on the targeted task rather than the HTM. Therefore, as demonstrated in Figure 6-3 and Figure 6-4, a number of different values ranging from 248 to 2048 for this parameter are tested to get the best value for the column count for the used dataset.

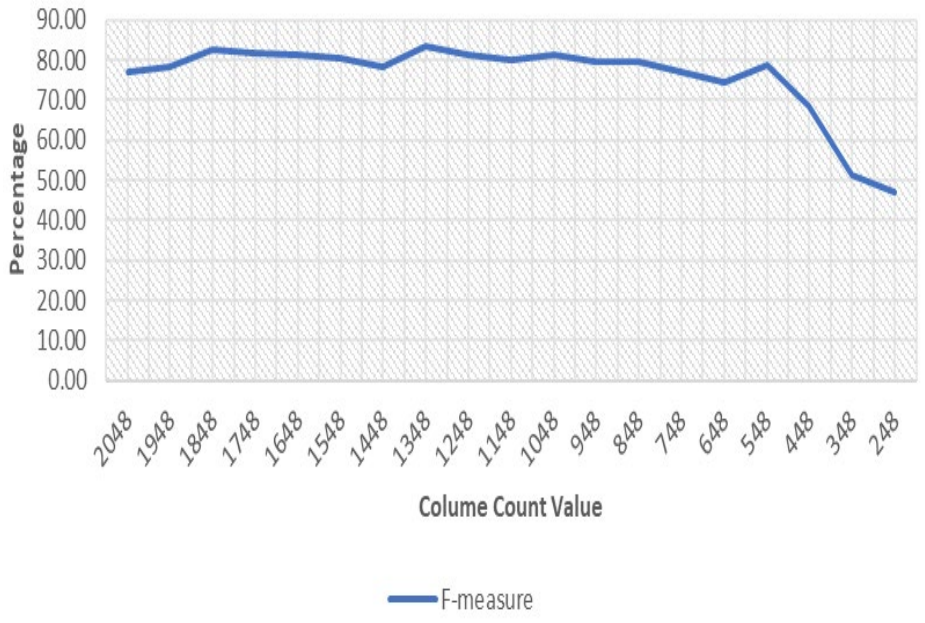


Figure 6-3: ColumnCount F-measure performance of SP and TM in the first level.

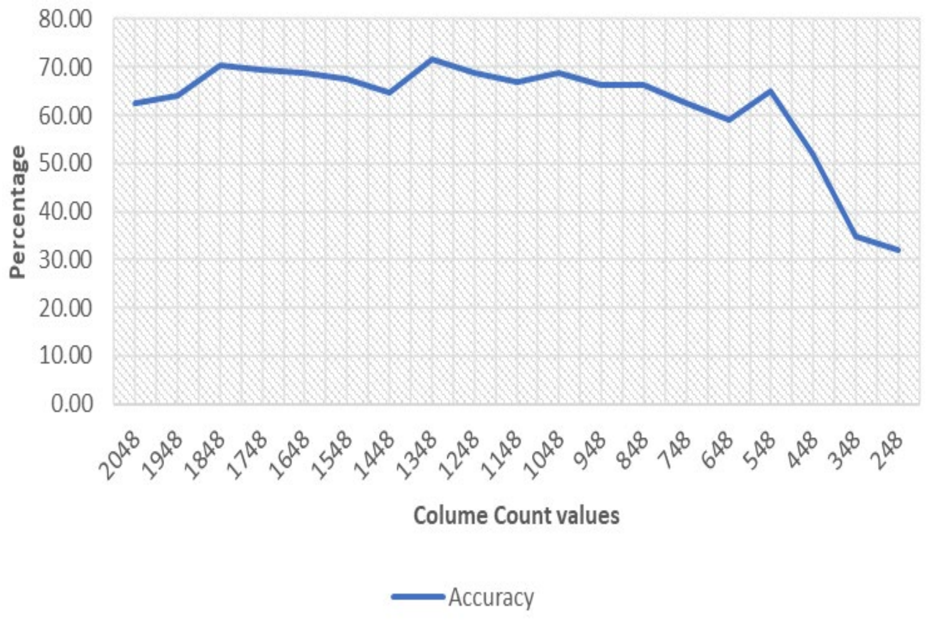


Figure 6-4: ColumnCount accuracy performance of SP and TM in the first level.

Figure 6-4 also shows the best performance when ColumnCount was set to 1348 with 72% accuracy and 83% f-measure. Comparing these values with the default value of 2048 recommended by NuPIC, the proposed method showed a rise in both accuracy by 10% and F-measure by 6.5% at the first level.

Each column has a natural centre over the input region, and the permanence values have a bias higher value at this location. Therefore, based on the nature of the input data, the best value for the ColumnCount was 1348.

Furthermore, the performance at the second level was also enhanced compared with the default value recommended by NuPIC. These enhancements in accuracy and f-measure are illustrated in Figure 6-5 and Figure 6-6, where accuracy rose by 7.7% when the value of ColumnCount was set to 1348, whereas f-measure was higher by

9.8%.

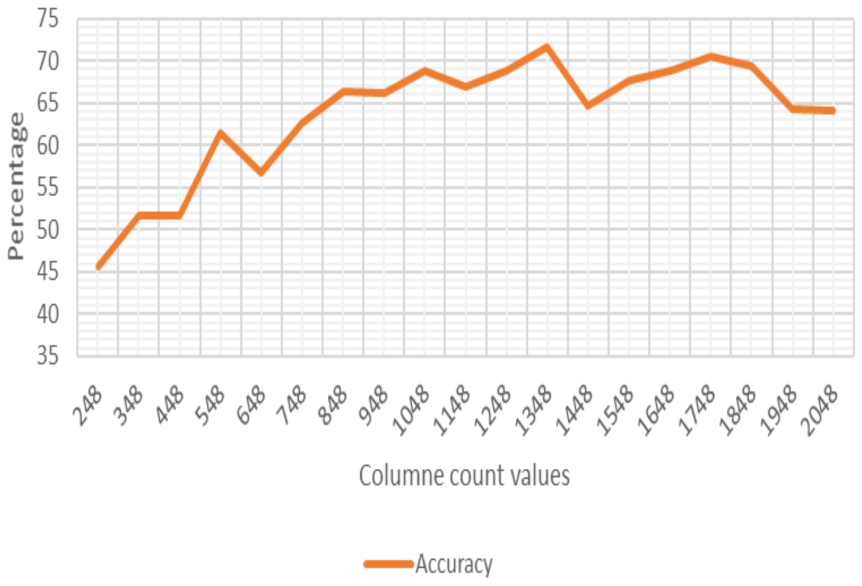


Figure 6-5: ColumnCount accuracy of SP and TM in the second level.

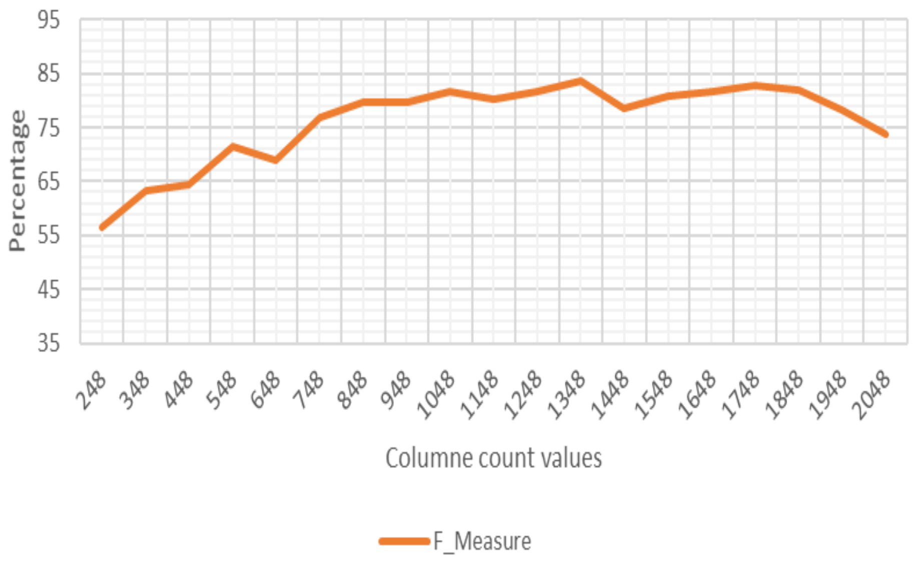


Figure 6-6: ColumnCount F-measure performance of SP and TM in the second level.

The performance of the framework is also tested in the second phase to fine tune the (numActiveColumnsPerInhArea) parameter. According to Hawkins et al.(2017), the importance of numActiveColumnsPerInhArea parameter relies on its ability to control the density of the activated column. Furthermore, if it is too high, it will lead to high false positives errors. When setting it too low, high false negatives will occur as it will be more difficult to detect patterns with low activation levels.

Because there were two levels of anomaly detection, the same process for testing ColumnCount was performed for numActiveColumnsPerInhArea. In the first level, different values were tested and performances were compared to the value recommended by NuPIC. As shown in Figure 6-7 and Figure 6-8, the best model performance for the model occurred when numActiveColumnsPerInhArea was set to 40 in the first level.

The same settings were tested in the second level, as illustrated in Figure 6-9 and Figure 6-10. The best performance in this case was found at numActiveColumnsPerInhArea set to four. Notably, there was a sharp decrease in performance when the value was set to 36; then, it steadily rose afterwards. Hence, the false negative rate was high enough to make it difficult to detect patterns with low activation levels.

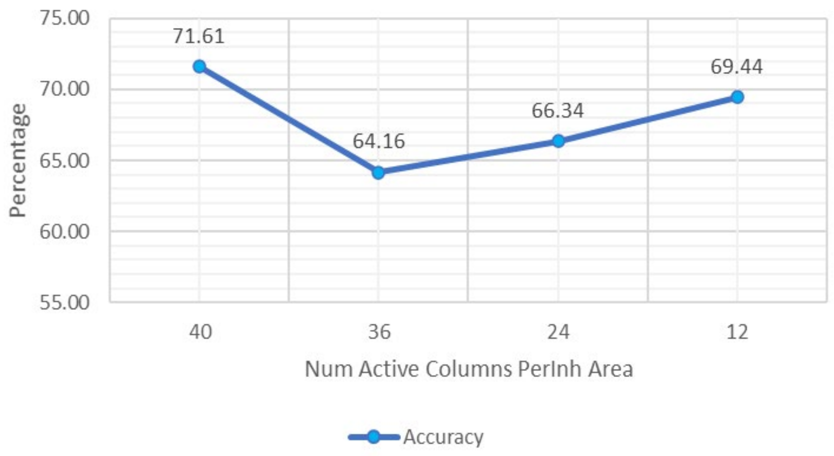


Figure 6-7: NumActiveColumn Accuracy performance of SP and TM in the first level.

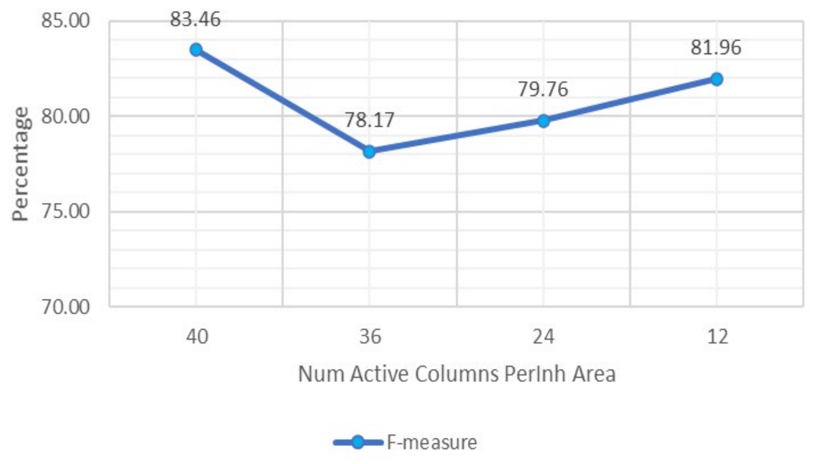


Figure 6-8: NumActiveColumn F-measure performance of SP and TM in the first level.

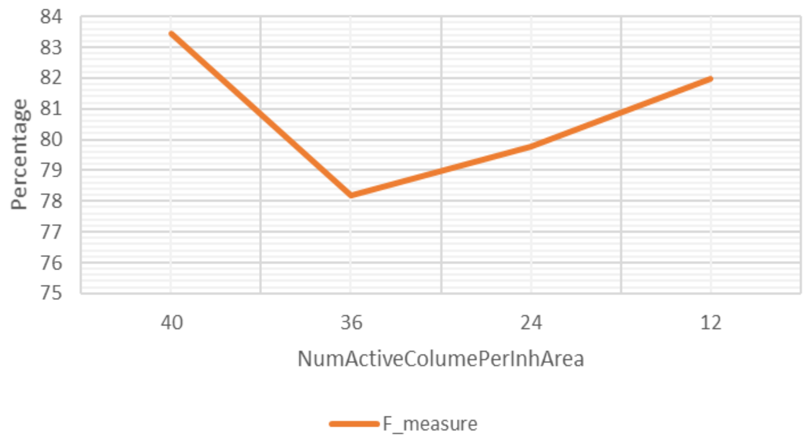


Figure 6-9: NumActiveColumn Accuracy performance of SP and TM in the second level

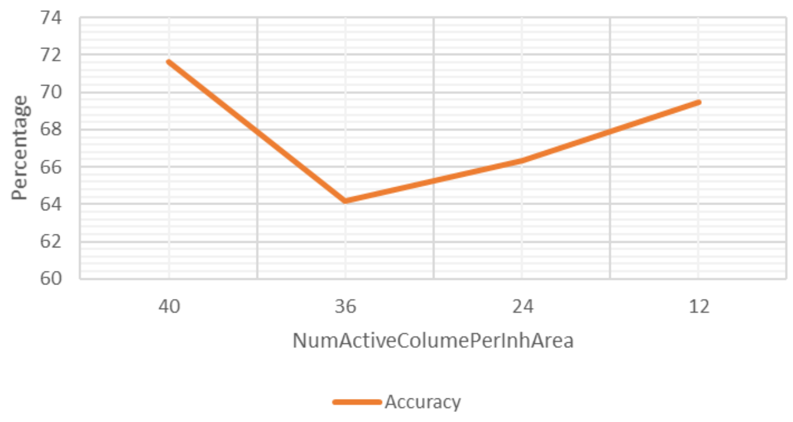


Figure 6-10: NumActiveColumn F-measure performance of SP and TM in the second level.

Performance was also challenged by testing different values of MaxBoot, which represented the third phase of parameter optimisation experiments. This parameter is used to control the strength of boosting. Furthermore, it relates to having the same active-duty cycles as its next column, which can be beneficial to SP. According to (Bastaki, 2019), using too much boosting can negatively affect the stability of SP outputs.

Experiments showed that the model reached its best performance when MaxBoot was set to three, as illustrated in Figure 6-11 and Figure 6-12 for the first level of the framework, with around 72% accuracy and an 83% f-measure, compared with the default value suggested by Numenta.

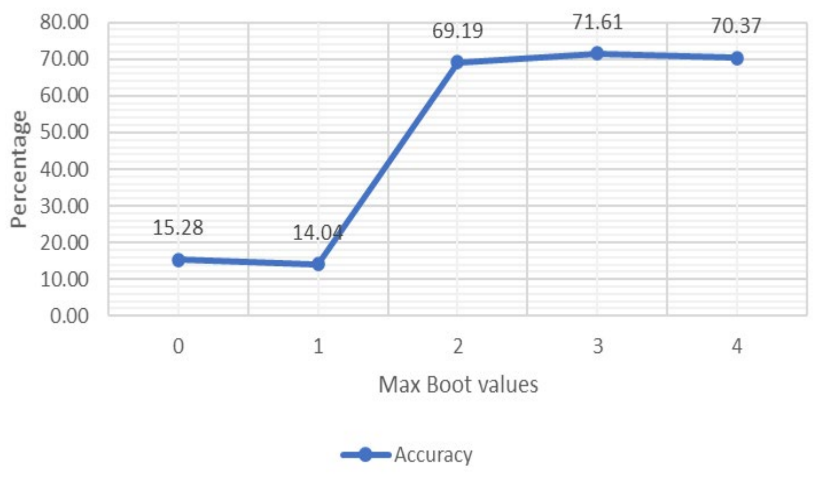


Figure 6-11: MaxBoot accuracy performance of SP in the first level.

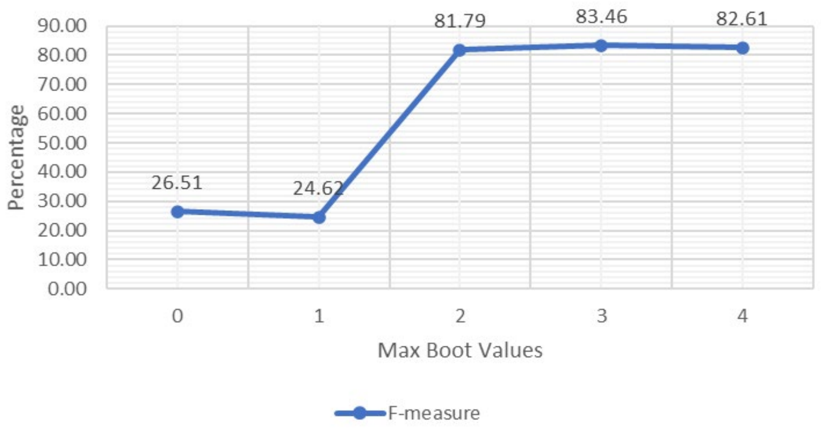


Figure 6-12: MaxBoot F-measure performance of SP in the first level.

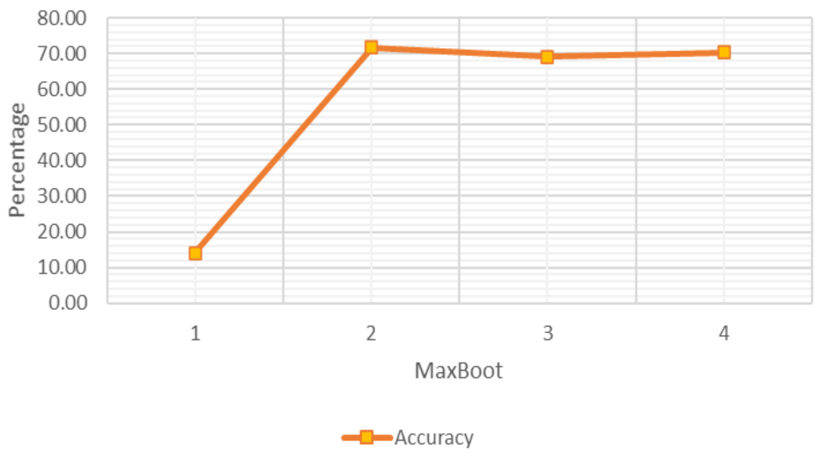


Figure 6-13: MaxBoot accuracy performance of SP in the second level.

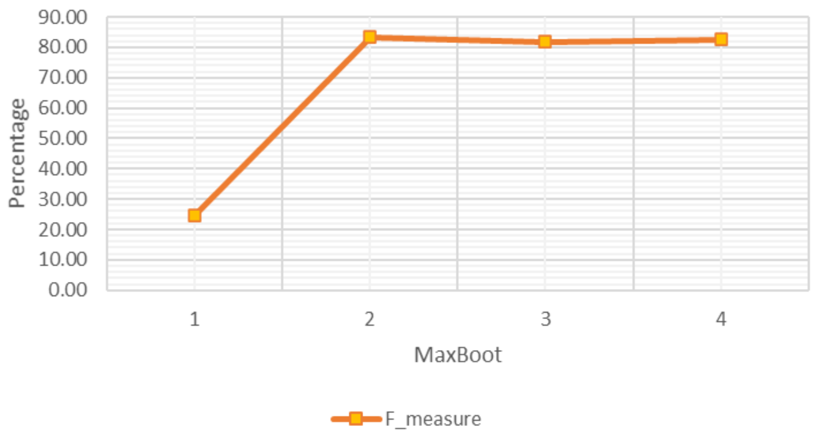


Figure 6-14: MaxBoot f-measure performance of SP in the second level

At the second level, the same experiments were performed for MaxBoost. Unlike the first level, as seen in Figure 6-13 and Figure 6-14, the model reaches its best performance when the MaxBoot sets to 2 with 71% accuracy and an 81% f-measure, respectively.

The outcomes of these experiments are summarised in Table 6-1, which were used in subsequent testing to deliver more accurate anomaly detection results than with the original SP and TM parameter settings to deliver more accurate performance in anomaly detection.

Table 6-1: SP and TM parameter settings used in experiments.

Parameter Name	SP	TM
numActiveColumnsPerInhArea	40	40
MaxBoot	3	3
ColumnCount	1348	1348

6.4 The Novel Multi-Level Anomaly Detection with Prediction Feedback Framework Experiments

Among the main contributions of this thesis is the development of a framework that provides better traffic anomaly detection and prediction. This framework is based on the one discussed earlier in Chapter 5. It is novel in that it predicts anomalies based on feedback from past detections. The proposed model is illustrated in Figure 6-15.

The framework allows for multistep prediction using the detected anomalies. Essentially, this move aims to improve prediction performance for traffic management systems. Basically, knowing that there is a potential road delay caused by an abnormal state will help models predict the future state of the road. Based on the research gap stated in Section 2.8, the early detection and prediction of abnormalities in roads are vital to smart transportation congestion management systems. To address this issue, an implementation of multi-level anomaly detection with a prediction feedback framework is proposed in this Chapter.

The steps used to design and implement this framework are summarised below. The framework presented in Chapter 5 was improved by adding a predictive model after concatenating the two levels of anomaly detection. The inputs that were fed to the prediction model were the anomalies and congestion metrics from previous levels. Afterward, the encoder encoded these inputs to form Sparse Distributed Representations (SDRs), and the same steps that were listed in Section 5.5 are repeated

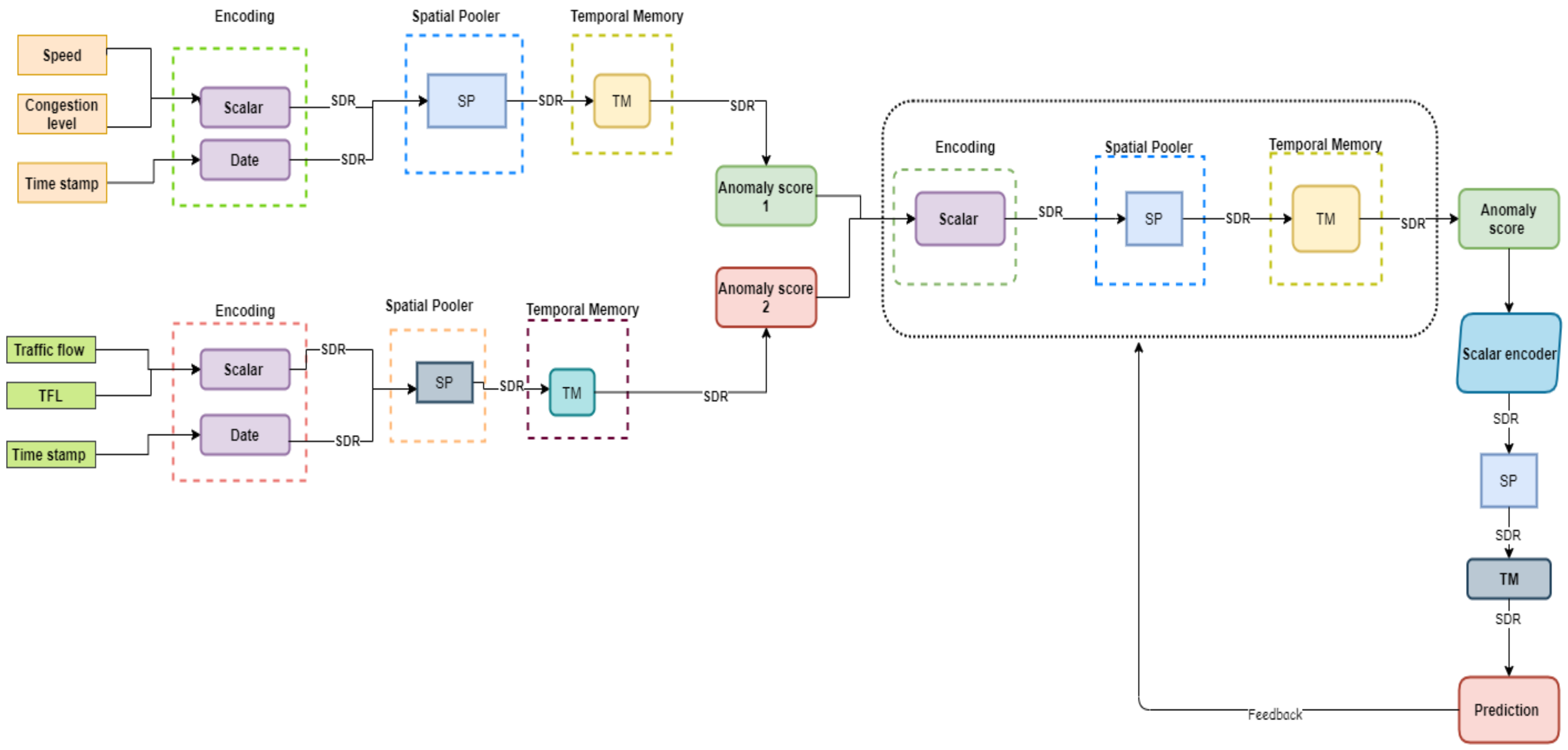


Figure 6-15: Detailed workflow of the proposed framework.

6.4.1 Context

The rationale behind this framework relies on the HTM capabilities that were identified in Section 3.7. Noting that HTM has its origins in neuroscience such that it mimics human cognitive learning functions, it was designed to use previous knowledge to learn new insights about objects and environments (Agrawal & Franklin, 2014). To the author's knowledge, no other publication has applied a feedbackward information flow in HTM, which is a challenging task, according to Numenta's latest research publications (Hole & Ahmad, 2021).

In the context of using HTM for smart transportation anomaly detection, the proposed framework must be able to predict possible anomalies based on the previous predictive model in the framework; then, this knowledge is fed back to the previous anomaly levels to identify similar patterns from the new data streams, which will ostensibly improve anomaly detection performance.

The next set of experiments proceeded in three stages: multi-level anomaly detection with a prediction model that provides feedback to the concatenated anomalies, conventional anomaly detection machine learning methods and evaluation based on recall, precision, accuracy and f-measure.

6.4.2 Data Preparation

For this part of the implementation, the datasets were divided into two sections: the first one is the datasets that were used in the anomaly detection experiments as was explained in Section 5.5.2. The second one is the aggregated anomaly datasets to be concatenated to get the predictions, which include the datasets that have been prepared for the anomaly experiments with the anoamliy score that will be feedbackwared to the model.

6.4.3 Experiment Settings

In this stage, the same experimental settings used in Section 5.5.3 are repeated using the same Inputs used to impelemnt the multi-level anomaly detection framework in Chapter 5 are used. Specifically, the first-level model was fed with Speed, Congestion Label, and Time of the Day data. Then the model was trained to obtain anomaly scores. Second, the model was fed Traffic Flow, Traffic Flow Label, and Time of the Day data. Then, it was again trained to obtain anomaly scores. Later, the outcomes of the first and second experiments were combined into one model. The next step entailed the predicted anomalies and was carried out using four multisteps followed by training. Then, the predictions were fed back to the previous anomaly detection level as described. Finally,, the performance evaluation metrics for the model were calculated.

6.4.4 Results

The results described in this section are the results of the final framework experiments used to evaluate the performance of the of the proposed multi-level anomaly detection with a prediction feedback framework. Following the individual anomaly experiments and their concatenations, a prediction model with a feedback loop was introduced. The performance evaluation metrics are given in Table 6-2, and Table 6-3 for the anomaly detection levels, and the prediction model part with the feedback. The results were rounded up to the nearest two decimal points.

As can be seen from Table 6-2 that the performance for each level in the framework different metrecs values were granted. This is due to dealing with different input data, which was described earlier in the experemnt settings.

Table 6-2: Performance evaluation metrics for the anomaly levels of the framework

	Accuracy	Precision	Recall	F-measure
Performance for the first level	62.5	85.3	70.1	76.9
Performance for the second level	64.0	71.3	40.5	73.6

Predictions were made from the resulting anomaly detection results. This step was carried out in a four multistep fashion. The model was allowed to learn and trained from the input data. After that, the performance evaluation metrics for the model were calculated.

To test the ability of the created model to predict traffic within a given period of time, predictions were set to take place every 15, 30, 45, and 60 minute intervals, labelled from 1 to 4 consecutively. The prediction results were then used to calculate the performance evaluation metrics for each prediction interval. Table 6-3 summarises these values.

Table 6-3: Performance evaluation metrics for the prediction model at four time intervals

	1 (15 minutes)	2 (30 minutes)	3 (45 minutes)	4 (60 minutes)
Accuracy	87.03	86.71	86.85	86.92
Precision	80.88	80.13	80.55	80.66
Recall	87.03	86.71	86.85	86.92
F-measure	82.38	81.89	81.97	82.20

Table 6-3 also shows that the model performed best when it made predictions at the shortest interval of 15 minutes. All four performance evaluation metrics were consistently higher for the 15-minute interval prediction than they are for any of the other longer intervals. Actually, performance evaluation metrics seem to be inversely proportional to prediction interval length. The longer the interval, the weaker the performance, and vice versa.

Finally, the predictive model is feeding back the results to the previous anomaly level, which is concatenated to identify the anomlies that comes from upper and lower level in the framework. The result are listed in Table 6-4 below.

Table 6-4: Performance evaluation metrics for the concatenated feedback

Accuracy	**Precision**	**Recall**	**F-measure**
92.4	44.4	92.1	61.0

The results show improvements in the performance compared to the results listed in Table 6-2. There was a 29% increase in the Accuracy rate of the combined model with feedback, which scored 92%, over the first anomaly model level as listed in Table 6-2. The Precision score of the concatenated feedback part of the framework seems to drop to almost half of the score of the anomaly level models. This sharp decrement might be due to the early detection of anomalies during the running of the previous anomaly parts before the prediction, which has led to a higher rate of accuracy, thus a lower rate of the Precision score of the concatenated model. In contrast, the Recall score of the combined feedback part is considerably higher than those of the previous level parts before the prediction. As the F-measure is affected by the values of the Precision and Recall score, it should come as no surprise that the F-measure value of the concatenated part, which is 61%, is lower than the anomaly parts without prediction feedback.

6.5 Evaluation

For the purposes of evaluation, the proposed framework was compared to the K-Nearest Neighbour Global Anomaly Score (KNN-GAS), the Independent Component Analysis-Local Outlier Probability (ICA-LoOP), and the Singular Value Decomposition Influence Outlier (SVD-IO) based on their respective evaluation metrics. The presentation of the evaluation of each model follows the following structure. For each model, a brief overview is given alongside a RapidMiner-extracted model workflow below. Next, the test plan that has been executed for each model is presented. After that, the RapidMiner test settings are described. Next, the calculated performance evaluation metrics for each model are given. Finally, the overall results for all four algorithms are presented and plotted together in a bar char for better visualisation of the differences in the scores of the metrics.

6.5.1 Proposed Framework

As stated in section 6.4, the proposed multi-level anomaly detection with prediction feedback framework was implemented and tested using the best parameter settings following the test plan below.

Test Plan:

1. *Input dataset selection*
2. *Build the model using NuPIC*
3. *Train the model*

4. *Test the model*
5. *Extract the performance metrics*

Proposed Framework Settings

The specific HTM parameter settings deployed for the proposed framework are based onthe experemnts conducted earlier on SP and TM settings that was run on the dataset, which is listed in Table 6-5 for the spatial pooler settings.

Table 6-5: SP settings.

Parameter	Description	Used value
columnCount	Total number of columns in the SP.	1348
globalInhibition	In the inhibition stage, the winning columns are chosen entirely from the region with the most active columns when set TRUE. Otherwise, the winning columns are selected based on their local neighbourhoods. Using global inhibition boosted performance x60.	1
Maxboost	Used to control the strength of boosting, which encourages columns to have similar activeDutyCycles as neighbours, which leads to more efficient column use. However, too much boosting may lead to SP output instability.	3
numActiveColumnsPerInhArea	Used to control the density of active columns. When specified, the localAreaDensity must be less than zero, and vice versa.	40
potentialPct	Percent of inputs within a column's potential radius that can be connected. If set to one, the column will be connected to every input within its potential radius.	0.8
seed	Seed for pseudo-random number generator.	1956
spVerbosity	SP diagnostic output verbosity control.	0
spatialImp	Spatial pooler implementation. Options: 'py' (slow, good for debugging) and 'cpp' (optimized).	cpp
synPermActiveInc	Amount by which an active synapse is incremented each round, specified as the percent of a fully grown synapse.	0.05
synPermConnected	Represents the connected synapse if the	0.1

Parameter	Description	Used value
	stability value is above the connected threshold.	
synPermInactiveDec	Amount by which an inactive synapse is decremented each round, specified as the percent of a fully grown synapse.	0.0005

The temporal memory settings were selected after running several tests to get the optimum settings. These settings were compared to swarm-generated settings. The best setting outcomes are listed in Table 6-6.

Table 6-6: TM settings

Parameter	Description	Used value
columnCount	Total number of columns in the TM.	1348
seed	Seed for pseudo-random number generator.	1960
activationThreshold	Number of synapses that must be active to activate a segment.	16
cellsPerColumn	Number of cells allocated per column	32
globalDecay	Value to decrease permanences when the global decay process runs. Global decay will remove synapses if their permanence value reaches 0. It will also remove segments when they no longer have synapses.	0
maxSegmentsPerCell	Maximum number of segments per cell	128
maxSynapsesPerSegment	Maximum number of synapses per segment	32
minThreshold	Minimum number of active synapses for a segment to be considered during search for the best-matching segments	12
newSynapseCount	New synapse formation count	20
pamLength	Number of time steps remaining in ‘Pay Attention Mode’ after detecting that it reached the end of a learned sequence.	1
initialPerm	Initial permanence for newly created synapses.	0.21
verbosity	Controls the verbosity of the TM diagnostic output	0

The calculated performance evaluation metrics for the proposed multi-level anomaly detection with prediction feedback framework are given in Table 6-7 below.

Table 6-7:Performance evaluation metrics for the proposed framework

Accuracy	**Precision**	**Recall**	**F_measure**
92.4	44.4	92.1	61.0

6.5.2 K-Nearest Neighbor Global Anomaly Score (KNN-GAS)

In nearest-neighbour-based algorithms, the outlier score is, according to Ramaswamy et al. (2000), the average range of the nearest kth neighbour in a dataset. Likewise, the anomaly scores in the KNN Global Anomaly Score algorithm (GAS) are calculated using the nearest k neighbours. This algorithm allocates around 99% of the execution computing to all K neighbours. The remaining 1% is used for storage (Ibid). Using the latest version of the RapidMiner studio, the algorithm was trained and tested. The same dataset that was used in the framework experiments was also used for KNN-GAS training and testing. Figure 6-16 shows the workflow for the KNN-GAS model extracted from the RapidMiner.

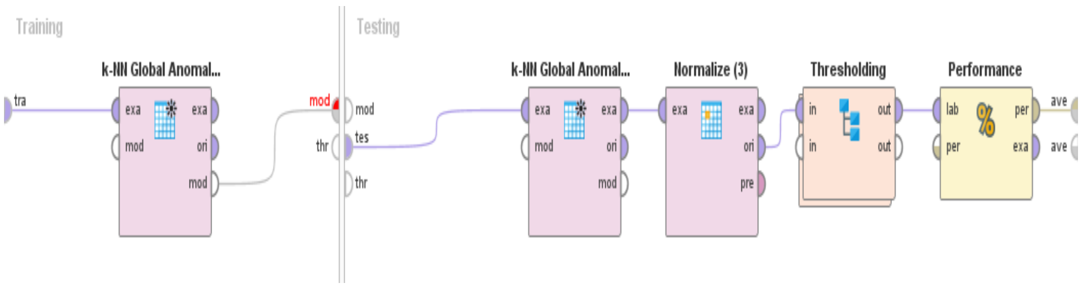


Figure 6-16: RapidMiner model for KNN-GAS.

Test Plan:

1. *Input dataset selection*
2. *Build the model using RapidMiner*

3. *Train the model*
4. *Test the model*
5. *Extract the performance metrics*

KNN-GAS RapidMiner Settings:

The RapidMiner test settings used to obtain the KNN-GAS are given in Table 6-8 below. For each selected parameter, a brief description is given, and the used value of the parameter is specified.

Table 6-8: RapidMiner test settings for KNN-GAS

Parameter	Description	Used value
k	Number of neighbours to be examined.	15
Measure Type	Type of measure to be used.	MixedMeasures
Mixed Measure	Select measure.	MixedEuclideanDistance

The model then yielded the anomaly scores which were then used to calculate its performance metrics. These are given in Table 6-9.

Table 6-9: Performance evaluation metrics for the KNN-GAS

Accuracy	Precision	Recall	F_measure
14.64%	14.79%	68.07%	25.54%

6.5.3 Independent Component Analysis-Local Outlier Probability (ICA-LoOP)

The Independent Component Analysis-Local Outlier Probability (ICA-LoOP) algorithm is used to find independent variables as linear variables. It is an improvement of the Principal Component Analysis (PCA) technique. The algorithm maximises the statistics to determine the independent components, whereas PCA primarily discovers unrelated components. Using the latest version of the RapidMiner studio, the ICA-LoOP algorithm was trained and tested. The same dataset that was used in the framework experiments was also used for ICA-LoOP training and testing. Figure 6-17 shows the workflow for the ICA-LoOP model extracted from the RapidMiner.

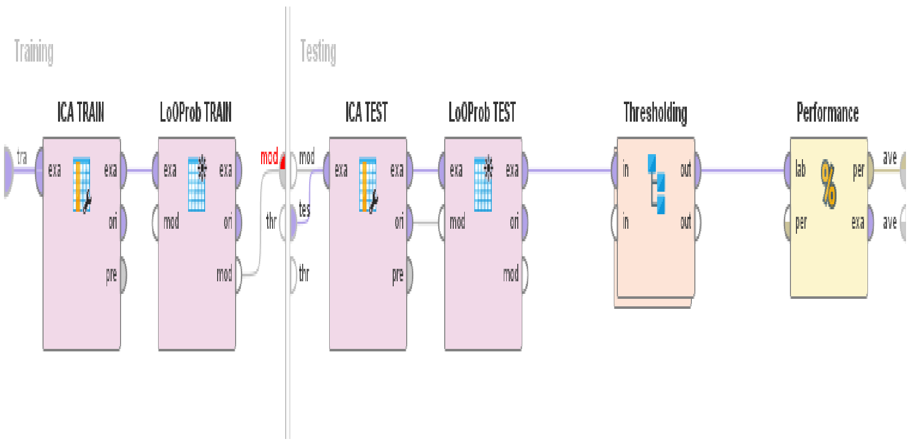


Figure 6-17: ICA-LoOP RapidMiner model.

Test Plan:

1. *Input dataset selection*
2. *Build the model using RapidMiner*
3. *Train the model*
4. *Test the model*
5. *Extract the performance metrics*

ICA-LoOP RapidMiner Settings:

The RapidMiner test settings for ICA-LoOP are given in Table 6-10 below. For each selected parameter, a brief description is given, and the used value of the parameter is specified.

Table 6-10: RapidMiner ICA-LoOP test settings

Parameter	Description	Used Value
Dimensionality Reduction	Type of dimensionality reduction in the number of attributes that should be deployed.	None
Algorithm Type	Type of algorithm that should be deployed.	Deflation
Function	Functional form of the G function that should be used to approximate neg-entropy.	Logcosh
k	Number of neighbours to be examined.	6
Normalization Factor	Standardises the outcome: the impact on results is weak.	3.0
Measure Type	Type of measure to be used.	MixedMeasures
Mixed Measure	This parameter is the select measure.	MixedEuclideanDistanc e
Parallelize Evaluation Process	Evaluation process that should be used to proceed in a parallel manner.	True
Number of Threads	Number of threads to be used for execution.	8

The model then produced the anomaly scores which were then used to calculate its performance metrics. These are given in Table 6-11.

Table 6-11:Performance evaluation metrics for the ICA-LoOP

Accuracy	Precision	Recall	F_measure
15.76%	15.48%	32.75%	26.75%

6.5.4 Singular Value Decomposition Influence Outlier (SVD-IO)

The Singular Value Decomposition Influence Outlier (SVD-IO) is perhaps one of the most widely used, with numerous applications in a variety of scientific disciplines. The algorithm is primarily used in matrix decomposition, statistics, as well as a data reduction technique. Based on SVD, the operator performs a dimensionality reduction of a given dataset. There is the option of either specifying the number of dimensions as needed or specifying a cumulative variance threshold. If a cumulative variance threshold is selected, components with cumulative variance above the specified threshold will be discarded. Using the latest version of the RapidMiner studio, the SVD-IO algorithm was trained and tested. The same dataset that was used in the framework experiments was also used for SVD-IO training and testing. Figure 6-18 shows the workflow for the SVD-IO model extracted from the RapidMiner.

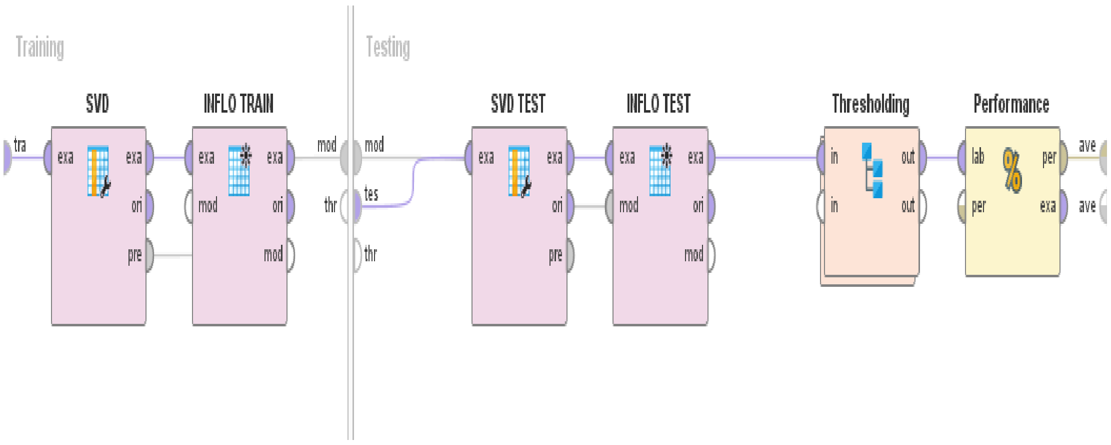


Figure 6-18: SVD-IO RapidMiner model.

Test Plan:

1. *Input dataset selection*
2. *Build the model using RapidMiner*
3. *Train the model*
4. *Test the model*
5. *Extract the performance metrics*

SVD-IO RapidMiner Settings:

The RapidMiner test settings for SVD are given in Table 6-12 below.

For each selected parameter, a brief description is given, and the used value of the parameter is specified.

Table 6-12: RapidMiner SVD test settings

Parameter	Description	Used Value
Dimensionality Reduction	This is about the type of dimensionality reduction in the number of attributes that ought to be deployed. By selecting 'fixed number', a fixed number of components will be kept. The dimension parameter determines the number of components that will be kept.	Fixed_number
Dimensions	This parameter appears only when Dimensionality Reduction is set to 'fixed number'. It is this parameter that determines the number of components that will be kept.	1

The model then yielded the anomaly scores which were then used to calculate its performance metrics. These are given in Table 6-13.

Table 6-13:Performance evaluation metrics for the SVD-IO

Accuracy	Precision	Recall	F_measure
15.90%	15.69%	67.17%	27.13%

6.5.5 Comparison Results

For better visualisation of the differences in the scores of the performance evaluation metrics of the four algorithms under investigation, the overall results for these algorithms are graphed in Figure 6-19. The bar chart plots the score values for each of the four algorithms on the four performance evaluation metrics that are reported in this study.

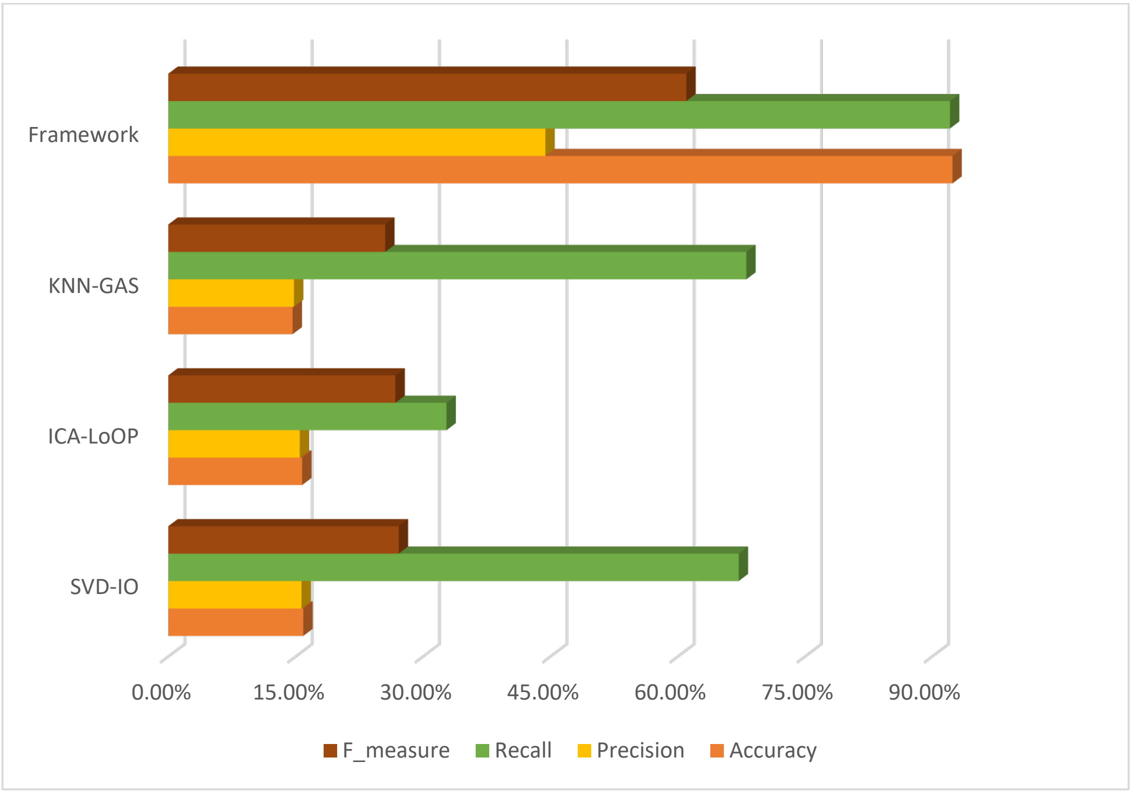


Figure 6-19: Performance comparison.

Overall, as the figure illustrates, the proposed framework consistently surpasses all three conventional methods on all the evaluation metrics. The two highest metric scores are Recall and Accuracy for the proposed HTM-based framework, with percentages above 90%. Conversely, the lowest metric scores are the Precision and Accuracy of the KNN-GAS, with percentages below 15%.

According to the Accuracy figures, the proposed framework outperforms all three conventional methods under evaluation. The difference is in the range of 77.76 to 76.5, in favour of the proposed framework, which scored an impressive 92.4%. The KNN-GAS achieved the lowest Accuracy of all algorithms, at 14.64.%. The other two conventional methods, the ICA-LoOP and the SVD-IO, achieved Accuracy values of 15.76% and 15.90%, respectively.

Comparison figures for Precision follow approximately the same pattern. However, the differences here have a smaller magnitude. The proposed framework still takes the lead, with a precision rate just a little below 45%. None of the conventional methods scored higher than 16%.

As mentioned above, Recall rates are among the highest in this implementation. The proposed framework has a rate of 92.1%, almost 25% higher than its closest conventional method competitor on this metric, which is the KNN-GAS algorithm. This result is to be contrasted with the Recall rates of the baseline experiments, which represented the odd-one-out case. It was the only time any conventional method had performed better than the proposed framework. The difference was small, however. the SVD-IO algorithm outperforms the proposed framework on this metric by just a little above 2%.

As far as F-measure values are concerned, the proposed algorithm, as expected given its performance on Recall and Precision metrics, scored higher than any of the conventional methods under evaluation. The difference is more than double, with the proposed framework scoring 61%, KNN-GAS 25.54%, ICA-LoOP 26.75%, and SVD-IO 27.13%.

6.6 Discussion

One of the main questions of this thesis revolves around whether HTM-based models are suitable for smart transport applications. The answer that this chapter suggests is *Yes*. The results of the set of experiments conducted in this study confirm that HTM-based algorithms are not only suitable for smart transport, but they also have real potential to improve the present status of traffic anomaly detection. A growing demand of transport management systems is to be context-aware: traffic events happen in a dynamic context, and they can easily multiply and worsen uncontrollably if they are not dealt with in a timely manner. As prevention is better than cure, early abnormality detection is a must in this case. Furthermore, early abnormality detection is just what the HTM-based multi-level anomaly detection with a prediction feedback framework proposed in this study offers.

The anomaly detection capacity of the proposed framework was put to the test in this chapter. A set of carefully designed experiments demonstrate that the proposed HTM-based framework performs consistently better than the three conventional methods investigated in the study on all the evaluation metrics used. With scores above 90%, the Recall and Accuracy of the proposed framework are the highest in the evaluation experiments.

Importantly, the difference in size between the Accuracy score of the proposed HTM-based framework, on the one hand, and the score of any of the conventional methods investigated, on the other, is the largest in the study. The difference is as large as 76% in favour of the proposed HTM-based framework. The conventional methods' Accuracy scores are below 16%, whereas the HTM-based framework scores above 92%.

Another equally important difference is between the performance of the proposed framework and that of the conventional methods along the F-measure metric. Here, the proposed HTM-based framework scored more than double the score of any of the conventional methods investigated. More specifically, the framework scores 61%, while the conventional methods' F-measure scores are below 28%.

These results could be attributed to the nature of the processes of learning and predicting in hierarchical structures. In the proposed framework, the hierarchical structure makes learning new patterns from the input streams easier to detect. At the same time, the hierarchical structure links similar patterns from each level of the hierarchy. Moreover, the ability of the regions to learn different patterns in a continuous way gives more strength to the framework to detect as early as possible any anomalies that occur. This capacity is not found in batch processing models. In other words, this feature of HTM-based framework results in a low precision but high recall, thus a higher accuracy of detecting abnormalities, as supported by the results in this chapter.

Among the conventional methods investigated in this study; KNN-GAS performed the best overall. However, it must be remembered that even with KNN-GAS, a small number of anomalous events still leads to moderate performance compared with the proposed framework. This result is in accordance which the findings of a comprehensive study conducted by Goldstein & Uchida (2016). Their study analysed the performance of several anomaly detection algorithms using ten datasets. Their datasets covered a wide variety of domains, including handwriting, breast cancer images, and language pronunciation. Their study revealed that KNN-GAS delivered more reliable results than the other algorithms they investigated. Note that none of the algorithms in the Goldstein & Uchida study (2016) were based on HTM.

The results of the multi-level anomaly detection part without the prediction feedback experiments also made it possible for the researcher to decide on which parameters and settings to use for the SP and TM. The choice of parameters was also useful during the later phase of algorithm training. In addition, parameter settings for the SP and TM were optimised using particle swarm optimisation to generate the best model parameters. This is an important step, because the performance of the algorithm heavily relies on the selected model parameters. Selecting the wrong parameters can negatively affect the performance of an algorithm, even if the algorithm itself is suitable to the task at hand (Lavesson & Davidsson, 2006; Domingos, 2012). In the current study, the model parameters were first swarm-optimised, and then subsequent adjustments were made to achieve the best possible accuracy improvement in the model.

The implications of the empirical findings of the study are considerable for the smart transport industry. The host of experiments run in this chapter demonstrate that the multi-level anomaly detection with prediction feedback framework is a realistically promising option to adopt for the optimisation of traffic detection and management. The results confirm both the suitability and potential of HTM to be harnessed for traffic detection data in general, and more specifically, for detecting abnormalities.

As highlighted in Chapter 2 of this thesis, congestion is a multi-faceted problem that requires multi-dimensional solutions. Unfortunately, the severity of the problem is on the rise, with far-reaching and grave repercussions for almost all aspects of our modern life. For example, global economies, the environment, and people’s health and emotional and physical well-being are all negatively affected by road congestion.

Technology can offer a lot to alleviate the congestion problem. One possible line of contribution is to predict congestion before it happens. Traffic anomaly detection is one promising way to go about managing road congestion. Detecting anomalies as early as possible will result in a flexible journey with less time spent on the road, thus lowering the environmental and health risks that come with traffic congestion. This thesis tested the accuracy of a novel HTM-based anomaly detection multi-level framework on traffic congestion data. The findings of the study open new possibilities for future testing and experimenting with more sophisticated setups in traffic congestion managements in smart transport.

Finally, it seems reasonable to claim that the potential of HTM theory is real, and it is vast. The capabilities of HTM-based algorithms should be leveraged to discover innovative solutions for traffic detection and management issues that are still unresolved. More research is needed to explore the full potential of this algorithm.

6.7 Summary

The performance of the proposed HTM-based multi-level anomaly detection with a pridection feedback framework was examined in this chapter.

At the beginning of this chapter, the initial experimental setup was explained. This Include identifying the design of the experimental work and the steps to test and evaluate the framework.

As a contribution to the field, this thesis determined the optimal parameter settings of SP and TM algorithms. This step was required to assess the performance of the proposed framework. Hence, several experiments were conducted in Section 6.3 to identify the most effective parameter values. The results showed improvements in framework performance compared with models that used original SP and TM settings.

In Section 6.4, the multi-level anomaly detection with prediction feedback framework was implemented using different settings to achieve the best performance. The implementation was conducted using the latest version of NuPIC, which is the platform code for implementing HTM-based models.

The performance of the proposed multi-level anomaly detection with prediction feedback framework was then evaluated against three conventional anomaly detection methods. These algorithms were (1) the K-Nearest Neighbour Global Anomaly Score (KNN-GAS), (2) the Independent Component Analysis-Local Outlier Probability (ICA-LoOP), and (3) the Singular Value Decomposition Influence Outlier (SVD-IO).

The evaluation presented in Section 6.5 demonstrated the superior performance of the proposed HTM-based anomaly detection multi-level framework compared with the KNN-GAS, the ICA-LoOP, and the SVD-IO. The F-measure results clearly revealed that the HTM-based anomaly detection multi-level framework was superior to its conventional methods competitors. The proposed framework achieved an F-score of 60%, thus outperforming the KNN-GAS by 35%, the ICA-LoOP by 34%, and the SVD-IO by 33%. These results provide evidence that the proposed HTM-based framework can better handle traffic anomaly detection than the three conventional anomaly detection methods. The implications of these findings to intelligent transport and the prospects they open for better-managed road traffic were also discussed in the chapter.

Chapter 7 Summary and Future Work

7.1 Summary

This chapter summarises the findings of this research and discusses future research avenues based on these findings.

The work presented in this thesis is a framework based on HTM theory to detect anomalies in road traffic data as part of a smart transport management application. The framework includes multi-level hierarchical parallel anomaly detection models. These models produce two anomaly scores. Each anomaly score accepts an SDR encoded from the input datasets and processes it through a spatial pooler and a temporal memory region. For simplicity, the two first regions were named A and B. When the two levels anomaly scores are produced, they become the input for a new, concatenating anomaly detector region named C. This region accepts the actual anomaly scores (not the SDRs) computed in the previous two regions, A and B. This step requires encoding for this new input. The new HTM region accepts a feedback connection from the last concatenated level to detect anomalies as soon as they occur. This step represents a top-down process feedback connection in the proposed framework, which improves the distribution of information throughout the HTM networks. To the best of the author's knowledge, this framework is the first implementation of a combination of multiple regions with a feedback loop HTM hierarchy. The performance of this framework was compared to the performance of some of the conventional anomaly detection techniques.

In the context of smart transport traffic management, there have been many attempts to provide solutions related to traffic detection. However, there is still a need for more feasible solutions, which motivated the researcher to propose this framework based on HTM theory. This research has attempted to answer the following questions:

RQ1 To what extent is the HTM theory suitable for smart transport applications?

RQ2 How do the features of HTM, such as the hierarchal structure and online learning, improve the performance of these applications?

RQ3 How is it possible to generate and prepare suitable data sets that can be applied to evaluate ML techniques and algorithms in smart transport scenarios?

Chapter 2 presents a literature review that informs the identification of the aims of this work and addresses objectives 1, 2, and 4. This review summarises IoT applications and technologies that could be used in transport applications. It then presents a comprehensive review of smart transport applications and their technologies. Previous studies on machine learning-based solutions were summarised. There has been considerable research on ML algorithms for smart transport applications. An intelligent transportation system consists of many smart processes that adopt a modular approach but work in harmony. Understanding vehicular traffic congestion is necessary for effective mobility and high-quality traffic management and safety systems. Recent research has produced several approaches to developing a sustainable smart transport system. Recently, various measures have improved performance with respect to criteria such as speed, travel time delay, and level of service (Falcocchio & Levinson, 2015). However, there is a need for a more reliable and accurate solution that is compatible with the continual changes in the data stream patterns that occur over time. This need can be satisfied using HTM as a solution.

Chapter 3 addresses RQ1 and objective 5 and includes information and a theoretical description of the HTM, which is a neuroscience-based algorithm that mimics the functionality of the neocortex in the human brain. The hierarchical structure of the HTM regions makes it a good choice for traffic detection applications. The roles of time and feedback in the HTM structure make it learn, infer, and predict quickly, which is crucial for intelligence transport applications. Furthermore, Chapter 3 also describes the implementation of the HTM theory by Numenta, which is known as the Cortical Learning Algorithms. CLA classifies the input patterns, and it will spot any different pattern. It first encodes the inputs using the proper encoder to the SDR. Next, the SP takes the encoded SDR to learn spatial patterns in the region column. Then, the SDR of the SP passes to the TM, which then can learn transition between spatial patterns. The results of the TM detect and classify abnormal patterns. However, there are some limitations of the HTM implementation, and the feedback level is not yet implemented, as explained in Chapter 6.

One of the objectives of this thesis research is addressed in Chapter 4, which includes a systematic review and analysis of existing datasets available for use in evaluating ML-based solutions. One of the issues facing smart transport research is the lack of standardised datasets for anomaly detection and classification tasks. To attempt to address this issue, the author used summarised criteria to identify suitable datasets to be used. One of these criteria was whether each dataset was available for download.

Published works from the past ten years were studied, as older datasets sometimes have too much missing content, because of the way they were obtained or the types of sensors used, which sometimes were unreliable.

The review of suitable datasets was structured to ignore results with terms unrelated to road traffic management, such as ‘network’, ‘network traffic,’ and ‘cloud’.

Based on the review criteria, a list of potential open datasets suitable for implementation of the proposed framework is presented in Chapter 4. The review revealed that many studies have used images as inputs. Therefore, the learning process could be slow because the model must first learn information that may appear trivial. The chapter concludes by identifying the dataset selected for use, which is provided by Highways England and can be found online under the Open Government Licence v3.0.

Chapter 5 introduced the design and implementation of the novel multi-level anoamly detection framework. The requirements and challenges of smart transport applications include the need for integrating and learning patterns from online data and dealing with changes in those streams. Other essential requirements include the ability to learn from historical data and predict future events. In addition, the challenges of implementing multi regions, , which are not yet implemented in the HTM, are addressed in this thesis. The chapter also contributes to the identification of the best parameter settings, based on several tests of the encoder settings.

Chapter 6 describes the multi-level anoamly detection with prediction feedback framework implementation. The chapter first presents several experiments conducted to optimise the SP and TM paramter settings to improve the framework. After that, prediction and feedback loop experiments were then conducted to improve the framework by adding the ability to predict traffic anomalies. These experiments addressed the research questions presented in Chapter 1 concerning whether the HTM theory can enhance the accuracy of anomaly detection in smart transport applications. The chapter also describes the pre-processing of the datasets judged to be suitable for this work.

In addition, Chapter 6 describes efforts to improve the performance by optimising the number of bits used to encode input values to produce SDRs fed into the HTM. Therefore, more experiments were conducted to determine how the model performs with different settings. The performance of the new settings was compared to the default ones. The results of these experiments show that the accuracy was 49.13% which improved by 2% compared to the defaults settings. Furthermore, the F-measure scores were around 59% which exceeded the original F-measure scores by 2.41%

As a secondary contribution to the knowledge, this thesis identifies the best parameter settings for both the spatial pooler and the temporal memory. After more experiments were performed, the accuracy rose by 10% and the F-measure by 6.5% for the SP and TM parameter settings.

As a secondary contribution to the knowledge, some conventional anomaly detection methods were compared with the proposed framework using the same datasets. The state-of-the-art anomaly detection algorithms compared to the proposed framework were the k-nearest-neighbours global anomaly score (KNN-GAS), the independent component analysis–local outlier probability (ICA-LoOP), and the singular value decomposition influence outlier (SVD-IO). The results reveal that the proposed framework achieved a 60% F-measure score, outperforming the KNN-GAS by 35%, the ICA-LoOP by 34%, and the SVD-IO by 33%. These results demonstrate that the proposed HTM-based framework performs better when dealing with traffic detection data.

7.2 Contribution to Knowledge

The contributions to knowledge by this thesis are as follows:

- The main contribution to knowledge is a novel multi-level anomaly detection framework based on HTM theory to detect anomalies in traffic. On the first level, anomalies are detected based on the congestion level in the training datasets. On the next level, the combined anomalies are fed to a prediction model to learn to predict anomalies in the future.
- The second major contribution to knowledge is to improve the learning capability of the proposed multi-level framework by introducing a feedback loop that can learn from patterns of data to detect anomalies in the context of smart transport traffic management.
- A secondary contribution of this work is a pre-processed subset of the Motorway Incident Detection and Automatic Signalling (MIDAS) traffic data used in the proposed framework. This contribution required a comprehensive review of available datasets for use in evaluating traffic congestion machine learning-based solutions and identifying the suitable set.
- Another secondary contribution to knowledge is to optimise the model parameters to identify the best parameter settings for the traffic congestion managements in smart transport.
- A performance evaluation of the proposed framework was conducted, and its

performance was compared to that of some conventional machine learning techniques for anomaly detection, using accuracy, recall, precision, and F-measure as performance metrics.

7.3 Conclusion and Future Work

One of the key parts of the HTM is the encoder, which translates the original input to binary SDRs. A basic encoder developed by Numenta handles inputs regardless of the scenario and the type of application domain. Furthermore, the numerous types of data in smart transport depend on the sensors and devices that produce them. Thus, more research could be done on the encoders to study their impact on precisely detecting anomalies. For example, a project could create a new encoder for the CLA that generates an SDR of various data types from multiple sensors and deals with high-dimensional binary and non-binary input data.

7.3.1 New Encoder

One of the key parts of the HTM is the encoder which translate the original input to a binary SDRs. There is basic type of encoders developed by Numenta to deal with inputs itself regardless the scenario and the type of application domain. Furthermore, because of the nature of the data in a smart transport that includes numerous types of data, depending on the type of sensors and devices that produced them. Thus, for a future work, more research could be done on the encoders to study its impacts on detecting anomalies precisely. Hence, a novel encoder could be created to be used to generates an SDR of different types of data that comes from deferent sensor types and to deal with high dimensional binary and non-binary input data.

7.3.2 Working on Datasets

In order to identify complex patterns of data streams in smart transport traffic management, further work is required to create datasets that can capture complex patterns of smart transport traffic management. They must include more factors to improve the capabilities of the data to evaluate the proposed framework to recognise new normal patterns. Concerning smart transport applications, the patterns change over time due to many factors, such as weather conditions, road maintenance, incidents, and speed restrictions that depend on the type of area (i.e., a freeway or urban area). Therefore, more data with various features are required to test the ability of the proposed framework to recognise the new normal patterns. As a future project, the framework will be tested using data collected from an urban area following the same methodology. This step will be a critical test that allows the framework to work in complex, real smart transport, and smart city applications. In addition to that, a future task is to explore the ability of the framework to learn new patterns that include unexpected factors, such as the weather forecast, to test the accuracy of the framework to predict and detect anomalies in more complex scenarios.

7.3.3 Testing the Proposed Framework with Different Domains

One of the design considerations of the proposed framework is generalisation, which means that this framework can be used not only in smart transport domain but can be used in other domains. The novelty of learning from previous knowledge, which is represented by the feedback loop in the framework, proved its capabilities in detecting anomalies. Furthermore, the HTM theory-based solutions were used in many other domains and confirmed their capabilities to improve results. With the spread of the

COVID-19 pandemic, which has a profound impact on the author, as a future project, the framework can be a tool for research related to COVID-19 tracking and symptoms recognition, which suggests a promising research contribution.

Bibliography

Abduljabbar, R., Dia, H., Liyanage, S. & Bagloee, S.A. (2019). Applications of artificial intelligence in transport: An overview. *Sustainability (Switzerland)*. 11 (1).

Aboah, A. (2021). A vision-based system for traffic anomaly detection using deep learning and decision trees. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

Afeefa, P.P. & Thulasidharan, P.P. (2018). Automatic License Plate Recognition(ALPR) using HTM cortical learning algorithm. *Proceedings of 2017 International Conference on Intelligent Computing and Control, I2C2 2017*. 2018-Janua. p.pp. 1–4.

Agrawal, P. & Franklin, S. (2014). Multi-layer Cortical Learning Algorithms. In: *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CCMB 2014: 2014 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain, Proceedings*. 23 January 2014, Institute of Electrical and Electronics Engineers Inc., pp. 141–147.

Ahmad, S. & Hawkins, J. (2015). *Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory*. [Online]. p.pp. 1–18. Available from: http://arxiv.org/abs/1503.07469.

Akhtar, M. & Moridpour, S. (2021). A review of traffic congestion prediction using artificial intelligence. *Journal of advanced transportation*. 2021. p.pp. 1–18.

Ali, I. & Asif, M. (2018). Applying security patterns for authorization of users in IoT based applications. *2018 International Conference on Engineering and Emerging Technologies, ICEET 2018*. 2018-January. p.pp. 1–5.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Alpaydın, E. (2014). *Introduction to machine learning*.

Alshammari, T., Alshammari, N., Sedky, M. & Howard (2019). *Biologically-Inspired Machine Intelligence Technique for Activity Classification in Smart Home Environments Evaluating Machine Learning Techniques for Activity Classification in Smart Home Environments*. [Online]. 12 (February). p.pp. 72–78. Available from: https://www.researchgate.net/publication/322978518. [Accessed: 12 May 2018].

Article, L.B., Bassel, G.W., Glaab, E., Marquez, J., Holdsworth, M.J. & Bacardit, J. (2011). *Functional Network Construction in Arabidopsis Using Rule-Based Machine Learning on Large-Scale Data Sets*. 23 (September). p.pp. 3101–3116.

Asghari, P., Rahmani, A.M. & Javadi, H.H.S. (2019). Internet of Things applications: A systematic review. *Computer Networks*. 148. p.pp. 241–261.

Asharf, J., Moustafa, N., Khurshid, H., Debie, E., Haider, W. & Wahab, A. (2020). A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics (Switzerland)*. 9 (7).

Atieno, O.P. (2009). An analysis of the strengths and limitation of qualitative and quantitative research paradigms. *Problems of Education in the 21st Century*. [Online]. 13. p.pp. 13–18. Available from: http://www.scientiasocialis.lt/pec/files/pdf/Atieno_Vol.13.pdf. [Accessed: 22

August 2018].

Atzori, L., Iera, A. & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*. [Online]. 54 (15). p.pp. 2787–2805. Available from: http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568.

Awad, M. & Khanna, R. (2015a). Cortical Algorithms. In: *Efficient Learning Machines*. [Online]. Berkeley, CA: Apress, pp. 149–165. Available from: https://doi.org/10.1007/978-1-4302-5990-9_8.

Awad, M. & Khanna, R. (2015b). *Efficient Learning Machines*. [Online]. Berkeley, CA: Apress. Available from: http://link.springer.com/10.1007/978-1-4302-5990-9.

Ayodele, O.T. (2010). Types of Machine Learning Algorithms. In: *New Advances in Machine Learning*. [Online]. InTech. Available from: https://cdn.intechopen.com/pdfs-wm/10694.pdf. [Accessed: 8 June 2017].

Bastaki, B.B. (2019). *Application of Hierarchical Temporal Memory to Anomaly Detection of Vital Signs for Ambient Assisted Living*.

Bekkers, J.M. (2011). Pyramidal neurons. *Current Biology*. [Online]. 21 (24). p.p. R975. Available from: http://dx.doi.org/10.1016/j.cub.2011.10.037.

Birks, M. & Mills, J. (2015). *GROUNDED THEORY*. Second edi. SAGE.

Blockeel, H., Borgwardt, K., Raedt, L. De, Domingos, P., Kersting, K. & Yan, X. (2011). *Guest editorial to the special issue on inductive logic programming , mining and learning in graphs and statistical relational learning*. p.pp. 133–135.

Bryman, A. (2016). *Social research methods*. [Online]. Oxford University Press.

Available from: https://global.oup.com/academic/product/social-research-methods-9780199689453?cc=gb&lang=en&.

Campagner, A., Ciucci, D., Svensson, C.-M., Figge, M.T. & Cabitza, F. (2021). Ground truthing from multi-rater labeling with three-way decision and possibility theory. *Information Sciences*. [Online]. 545. p.pp. 771–790. Available from: https://www.sciencedirect.com/science/article/pii/S0020025520309609.

ÇETİNKAYA, Z. & HORASAN, F. (2021). Decision Trees in Large Data Sets. *International Journal of Engineering Research and Development*. [Online]. 13 (1). p.pp. 140–151. Available from: https://dergipark.org.tr/en/doi/10.29137/umagd.763490.

Chakravarthy, H., Bachan, P., Roshini, P. & Chandrasekharan, R.K. (2012). *Network and Complex Systems Bio Inspired Approach as a Problem Solving Technique*. [Online]. 2 (2). Available from: www.iiste.org. [Accessed: 8 February 2018].

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection: A survey. *ACM Reference Format*. [Online]. 41 (15). Available from: http://doi.acm.org/10.1145/1541880.1541882. [Accessed: 24 June 2021].

Chang, H., Lee, Y., Yoon, B. & Baek, S. (2012). Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences. *IET Intelligent Transport Systems*. [Online]. 6 (3). p.p. 292. Available from: http://digital-library.theiet.org/content/journals/10.1049/iet-its.2011.0123.

Chen, C., Seo, H., Jun, C.H. & Zhao, Y. (2022). Pavement crack detection and classification based on fusion feature of LBP and PCA with SVM. *International Journal of Pavement Engineering*. [Online]. 23 (9). p.pp. 3274–3283. Available from: https://www.tandfonline.com/doi/abs/10.1080/10298436.2021.1888092. [Accessed: 28 September 2022].

Chen, X., Wang, W. & Li, W. (2012). An overview of Hierarchical Temporal Memory: A new neocortex algorithm. *Proceedings of 2012 International Conference on Modelling, Identification and Control, ICMIC 2012*. p.pp. 1004–1010.

Chen Xi, Wang Wei & Li Wei (2012). *Proceedings of 2012 International Conference on Modelling, Identification and Control : Wuhan, China, June 24th-26th, 2012.* IEEE.

Chiabaut, N. & Faitout, R. (2021). Traffic congestion and travel time prediction based on historical congestion maps and identification of consensual days. *Transportation Research Part C: Emerging Technologies*. 124. p.p. 10292.

Chin, J., Callaghan, V. & Ben Allouch, S. (2019). The Internet-of-Things: Reflections on the past, present and future from a user-centered and smart environment perspective. *Journal of Ambient Intelligence and Smart Environments*. 11 (1). p.pp. 45–69.

Chruścik, A., Kauter, K., Whiteside, E. & Windus, L. (2021). *Fundamentals of Anatomy and Physiology*. Australian. Place of publication not identified: University of Southern Queensland.

Cook, A.A., Misirli, G. & Fan, Z. (2020). Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*. [Online]. 7 (7) p.pp. 6481–6494. Available from: http://www.ieee.org/publications_standards/publications/rights/index.html. [Accessed: 29 June 2021].

Cui, Y., Ahmad, S. & Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*. [Online]. 28 (11). p.pp. 2474–2504. Available from: http://www.mitpressjournals.org/doi/10.1162/NECO_a_00893. [Accessed: 8 February 2018].

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

Davies, M. & Hughes, N. (2014). *Doing a successful research project : using qualitative or quantitative methods*. 2nd Ed. Palgrave Macmillan.

Devijver, P.A. (2020). Statistical Pattern Recognition. In: *Pattern Recognition*. pp. 14–36.

Dogra, A.K. & Kaur, J. (2022). Moving towards smart transportation with machine learning and Internet of Things (IoT): a review. *Journal of Smart Environments and Green Computing*. [Online]. 2 (1). p.pp. 3–18. Available from: https://segcjournal.com/article/view/4642. [Accessed: 17 June 2022].

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*. [Online]. 55 (10) p.pp. 78–87. Available from: https://dl.acm.org/doi/abs/10.1145/2347736.2347755. [Accessed: 9 August 2021].

Domínguez-Barbero, D., García-González, J., Sanz-Bobi, M.A. & Sánchez-Úbeda, E.F. (2020). Optimising a microgrid system by deep reinforcement learning techniques. *Energies*. [Online]. 13 (11). p.p. 2830. Available from: https://www.mdpi.com/1996-1073/13/11/2830/htm. [Accessed: 16 August

2021].

Du, K.-L. & Swamy, M.N.S. (2014). Fundamentals of Machine Learning. In: *Neural Networks and Statistical Learning*. [Online]. London: Springer London, pp. 15–65. Available from: http://link.springer.com/10.1007/978-1-4471-5571-3_2. [Accessed: 8 February 2018].

Eli-Chukwu, N.C. (2019). Applications of Artificial Intelligence in Agriculture: A Review. *Engineering, Technology & Applied Science Research*. 9 (4). p.pp. 4377–4383.

Elie Bursztein (2022). *Security and Privacy Group Deep-Cryptanalysis Fashion or Revolution? with the help of many Googlers*. [Online]. Available from: https://elie.net/deep-crypto. [Accessed: 21 September 2022].

Elmustafa, S.A.A. & Kamal, Z. (2017). Internet of Things Applications, Challenges and Related Future Technologies. *world scientific news*.

van Engelen, J.E. & Hoos, H.H. (2020). A survey on semi-supervised learning. *Machine Learning*. [Online]. 109 (2). p.pp. 373–440. Available from: https://link.springer.com/article/10.1007/s10994-019-05855-6. [Accessed: 20 September 2022].

Erdelyi, J.F., Amblard, F., Gaudou, B., Kaddoum, E. & Verstaevel, N. (2022). *Exploration of Model Coupling Strategies in a Hybrid Agent-Based Traffic Simulation*. Cham: International Workshop on Multi-Agent Systems and Agent-Based Simulation. Springer.

Falcocchio, J.C. & Levinson, H.S. (2015). Measuring traffic congestion. *Springer Tracts on Transportation and Traffic*. 7 (January 1998). p.pp. 93–110.

Farizawani, A.G., Puteh, M., Marina, Y. & Rivaie, A. (2020). A review of artificial neural network learning rule based on multiple variant of conjugate gradient approaches. *Journal of Physics: Conference Series*. 1529 (2). p.p. 2.

Figueiras, P., Herga, Z., Guerreiro, G., Rosa, A., Costa, R. & Jardim-Goncalves, R. (2018). Real-Time Monitoring of Road Traffic Using Data Stream Mining. In: *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*. 13 August 2018, Institute of Electrical and Electronics Engineers Inc.

Firouzi, F., Farahani, B., Weinberger, M., DePace, G. & Aliee, F.S. (2020). Iot fundamentals: Definitions, architectures, challenges, and promises. In: *Intelligent Internet of Things*. [Online]. Springer, pp. 3–50. Available from: https://link.springer.com/chapter/10.1007/978-3-030-30367-9_1. [Accessed: 28 July 2021].

Fleming Sean (2019). *Traffic congestion cost the US economy nearly $87 billion in 2018 | World Economic Forum*. [Online]. Available from: https://www.weforum.org/agenda/2019/03/traffic-congestion-cost-the-us-economy-nearly-87-billion-in-2018/. [Accessed: 9 July 2021].

Frankenfiels, J. (2020). *Artificial neural network (ANN)*. [Online]. 2020. Available from: https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp. [Accessed: 2 August 2021].

Fu, Y., Guo, X., Xie, Y., Zhang, D. & Li, H. (2015). Disease Diagnosis Supported by Hierarchical Temporal Memory. In: *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and*

Communications and Its Associated Workshops (UIC-ATC-ScalCom). [Online]. August 2015, IEEE, pp. 863–870. Available from: http://ieeexplore.ieee.org/document/7518347/. [Accessed: 8 February 2018].

Garalevicius, S.J. (2007). Memory – Prediction Framework for Pattern Recognition : Performance and Suitability of the Bayesian Model of Visual Cortex. In: *FLAIRS Conference*. [Online]. 2007, pp. 92–97. Available from: https://www.aaai.org/Papers/FLAIRS/2007/Flairs07-018.pdf. [Accessed: 3 February 2018].

Gavrilova, Y. (2021). *Anomaly Detection in Machine Learning*. [Online]. Serokell Software Development Company. Serokell. Available at. Available from: https://serokell.io/blog/anomaly-detection-in-machine-learning.

Ge, B., Li, X., Jiang, X., Sun, Y. & Liu, T. (2018). A dictionary learning approach for signal sampling in task-based fmri for reduction of big data. *Frontiers in Neuroinformatics*. 12.

Goldstein, M. & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS ONE*. [Online]. 11 (4). p.p. e0152173. Available from: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173. [Accessed: 25 July 2021].

Goves, C., North, R., Johnston, R. & Fletcher, G. (2016a). Short Term Traffic Prediction on the UK Motorway Network Using Neural Networks. In: *Transportation Research Procedia*. [Online]. 1 January 2016, Elsevier, pp. 184–195. Available from: https://www.sciencedirect.com/science/article/pii/S2352146516300199.

[Accessed: 4 February 2018].

Goves, C., North, R., Johnston, R. & Fletcher, G. (2016b). Short Term Traffic Prediction on the UK Motorway Network Using Neural Networks. In: *Transportation Research Procedia*. [Online]. 1 January 2016, Elsevier, pp. 184–195. Available from: https://www.sciencedirect.com/science/article/pii/S2352146516300199. [Accessed: 8 February 2018].

Gubbi, J., Buyya, R. & Marusic, S. (2013). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*. 29 (7). p.pp. 1645–1660.

Haenlein, M. & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*. [Online]. 61 (4). p.pp. 5–14. Available from: https://journals.sagepub.com/doi/abs/10.1177/0008125619864925?journalCode=cmra. [Accessed: 11 August 2021].

Haldurai, L., Madhubala, T. & Rajalakshmi, R. (2016). A Study on Genetic Algorithm and its Applications. *International Journal of Computer Sciences and Engineering*. [Online]. 4 (10). p.pp. 139–143. Available from: www.ijcseonline.org.

Hawkins, J. & Ahmad, S. (2016). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Frontiers in Neural Circuits*. [Online]. 10. p.p. 23. Available from: http://journal.frontiersin.org/Article/10.3389/fncir.2016.00023/abstract. [Accessed: 8 February 2018].

Hawkins, J., Ahmad, S. & Dubinsky, D. (2010). *Hierarchical Temporal Memory including HTM Cortical Learning Algorithms*. [Online]. Available from: http://www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf.

Hawkins, J., Ahmad, S., Purdy, S. & Lavin, A. (2017). Initial online release 0.4. *Biological and Machine Intelligence (BAMI)*. [Online]. Available from: https://numenta.com/resources/biological-and-machine-intelligence/.

Hawkins, J. & Blakeslee, S. (2004). On Intelligence. *New York St. Martin's Griffin*. p.p. pp.156-8.

Hawkins, J. & George, D. (2006). *Hierarchical Temporal Memory--Concepts, Theory, and Terminology*.

Hilb, M. (2020). Toward artificial governance? The role of artificial intelligence in shaping the future of corporate governance. *Journal of Management and Governance*. [Online]. 24 (4). p.pp. 851–870. Available from: https://link.springer.com/article/10.1007/s10997-020-09519-9. [Accessed: 11 August 2021].

Hole, K.J. & Ahmad, S. (2021). A thousand brains: toward biologically constrained AI. *SN Applied Sciences*. [Online]. 3 (8) p.pp. 1–14. Available from: https://link.springer.com/article/10.1007/s42452-021-04715-0. [Accessed: 15 November 2022].

Hopkins, W.G. (2008). Research designs: choosing and fine-tuning a design for your study. *Sportscience*. 12 (1). p.pp. 1–3.

Hu, M., Wang, Y., Feng, X., Zhou, S., Wu, Z. & Qin, Y. (2022). *Robust Anomaly*

Detection for Time-series Data. [Online]. Available from: https://arxiv.org/abs/2202.02721v1. [Accessed: 26 September 2022].

Huang, M. (2022). SVM-Based Real-Time Identification Model of Dangerous Traffic Stream State X. Ning (ed.). *Wireless Communications and Mobile Computing*. [Online]. 2022. p.pp. 1–9. Available from: https://www.hindawi.com/journals/wcmc/2022/6260395/. [Accessed: 28 September 2022].

Jabbarpour, M.R., Jalooli, A., Shaghaghi, E., Noor, R.M., Rothkrantz, L., Khokhar, R.H. & Anuar, N.B. (2014). Ant-based vehicle congestion avoidance system using vehicular networks. *Engineering Applications of Artificial Intelligence*. [Online]. 36. p.pp. 303–319. Available from: https://www.sciencedirect.com/science/article/pii/S0952197614001997. [Accessed: 4 February 2018].

John, F.Z., Amr, A.E., Sherif, E.H., Sabry, F.S. & Fayez, F.A. (2016). Framework for Traffic Congestion Prediction. *Journal of Scientific and Engineering Research*. 7 (5). p.pp. 1205–1210.

Jr, J.O., Roque, A.C., Schiavo, F., Sguerra, B.M., Miyamoto, B.A., Mourão, F.A., Alves, T.Y. & Suiti, A.D.P. (2017). *Addressing the Golden Hour : A machine learning approach to improve emergency response time*.

Julio, N., Giesen, R. & Lizana, P. (2016). Real-time prediction of bus travel speeds using traffic shockwaves and machine learning algorithms. *Research in Transportation Economics*. [Online]. 59. p.pp. 250–257. Available from: http://dx.doi.org/10.1016/j.retrec.2016.07.019.

Kamble, S.J. & Kounte, M.R. (2020). Machine Learning Approach on Traffic

Congestion Monitoring System in Internet of Vehicles. *Procedia Computer Science*. [Online]. 171. p.pp. 2235–2241. Available from: https://www.sciencedirect.com/science/article/pii/S1877050920312321.

Kanade, V. (2022). *Linear Regression Types, Equation, Examples, Best Practices*. [Online]. 8 April 2022. spicework. Available from: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/. [Accessed: 22 September 2022].

Karuppanagounder, K. & Muneera, C.P. (2018). Economic impact of traffic congestion- estimation and challenges. *European Transport*. (68).

Kashyap, P. (2017). Let's Integrate with Machine Learning. *Machine Learning for Decision Makers*. [Online]. p.pp. 1–34. Available from: https://link.springer.com/chapter/10.1007/978-1-4842-2988-0_1. [Accessed: 24 July 2021].

Kaur, N., SAHDEV, S.L., SHARMA, M. & SIDDIQUI, L. (2020). Banking 4.0: "the Influence of Artificial Intelligence on the Banking Industry & How Ai Is Changing the Face of Modern Day Banks". *International Journal of Management*. 11 (6). p.pp. 577–585.

Khan, N.A. (2017). *Real Time Predictive Monitoring System for Urban Transport Real Time Predictive Monitoring System for Urban Transport Supervisory Team: Professor Vesna Brujic-Okretic, Professor Souheil Khaddaj Real Time Predictive Monitoring System for Urban Transport*. [Online]. Available from: http://eprints.kingston.ac.uk/id/eprint/39019/. [Accessed: 2 July 2021].

Khan, Z., Khan, S.M., Tine, J.M., Comert, A.T., Rice, D., Comert, G., Michalaka, D., Mwakalonge, J., Majumdar, R. & Chowdhury, M. (2021). arXiv preprint.

Hybrid Quantum-Classical Neural Network for Incident Detection. [Online]. Available from: http://arxiv.org/abs/2108.01127.

Kong, X., Xu, Z., Shen, G., Wang, J., Yang, Q. & Zhang, B. (2016). Urban traffic congestion estimation and prediction based on floating car trajectory data. *Future Generation Computer Systems*. 61. p.pp. 97–107.

Krajna, A., Brcic, M., Lipic, T. & Doncevic, J. (2022). Explainability in reinforcement learning: perspective and position. *arXiv preprint arXiv:2203.11547*. [Online]. 18 (9). Available from: http://arxiv.org/abs/2203.11547.

Kumar, K., Parida, M. & Katiyar, V.K. (2013). Short Term Traffic Flow Prediction for a Non Urban Highway Using Artificial Neural Network. *Procedia - Social and Behavioral Sciences*. 104. p.pp. 755–764.

Kumar, K., Parida, M. & Katiyar, V.K. (2015). Short term traffic flow prediction in heterogeneous condition using artificial neural network. *Transport*. [Online]. 30 (4). p.pp. 397–405. Available from: http://www.tandfonline.com/doi/full/10.3846/16484142.2013.818057.

Kumar, N. & Raubal, M. (2021). Applications of deep learning in congestion detection, prediction and alleviation: A survey. *Transportation Research Part C: Emerging Technologies*. [Online]. 133. p.p. 103432. Available from: http://arxiv.org/abs/2102.09759. [Accessed: 29 September 2022].

Kwoczek, S., Di Martino, S. & Nejdl, W. (2014). Predicting and visualizing traffic congestion in the presence of planned special events. *Journal of Visual Languages and Computing*. 25 (6). p.pp. 973–980.

Längkvist, M., Karlsson, L. & Loutfi, A. (2014). *A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling*. [Online]. Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:oru:diva-34597. [Accessed: 21 September 2022].

Lavesson, N. & Davidsson, P. (2006). Quantifying the impact of learning algorithm parameter tuning. In: *Proceedings of the National Conference on Artificial Intelligence*. [Online]. 2006, pp. 395–400. Available from: www.aaai.org. [Accessed: 9 August 2021].

Lee, T.J., Gottschlich, J., Tatbul, N., Metcalf, E. & Zdonik, S. (2018). *Precision and Recall for Range-Based Anomaly Detection*. [Online]. Available from: https://arxiv.org/abs/1801.03168/. [Accessed: 24 June 2021].

Lee, Y. & Min, O. (2017). Comparative Analysis of Machine Learning Algorithms to Urban Traffic Prediction. *Ieee*. p.pp. 1034–1036.

Li, B., Hou, B., Yu, W., Lu, X. & Yang, C. (2017). Applications of artificial intelligence in intelligent manufacturing: a review. *Frontiers of Information Technology & Electronic Engineering 2017 18:1*. [Online]. 18 (1). p.pp. 86–96. Available from: https://link-springer-com.sdl.idm.oclc.org/article/10.1631/FITEE.1601885. [Accessed: 12 August 2021].

Liang, Z., Chen, H., Song, Z., Zhou, Y. & Zhang, B. (2018). Traffic congestion incident detection and dissipation algorithm for urban intersection based on FCD. In: *2017 3rd IEEE International Conference on Computer and Communications, ICCC 2017*. [Online]. December 2018, IEEE, pp. 2578–2583. Available from: http://ieeexplore.ieee.org/document/8323001/. [Accessed: 12

May 2018].

Liu, C., Zhao, M., Sharma, A. & Sarkar, S. (2019). Traffic Dynamics Exploration and Incident Detection Using Spatiotemporal Graphical Modeling. *Journal of Big Data Analytics in Transportation*. [Online]. 1 (1). p.pp. 37–55. Available from: https://link.springer.com/article/10.1007/s42421-019-00003-x. [Accessed: 1 October 2022].

Liu, X., Liu, X., Wang, Y., Pu, J. & Zhang, X. (2016). Detecting anomaly in traffic flow from road similarity analysis. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 2016, Springer Verlag, pp. 92–104. Available from: https://link.springer.com/chapter/10.1007/978-3-319-39958-4_8. [Accessed: 24 June 2021].

Lutkevich, B. (2022). *what is a facility? Definition and examples*. [Online]. May 2022. Available from: https://www.techtarget.com/searchunifiedcommunications/definition/real-time-application-RTA. [Accessed: 30 June 2022].

Mallik, S. (2014). Intelligent Transportation System. *International Journal of Civil Engineering Research*. [Online]. 5 (4). p.pp. 2278–3652. Available from: http://www.ripublication.com/ijcer.htm.

Marques, H.O., Campello, R.J.G.B., Sander, J. & Zimek, A. (2020). Internal Evaluation of Unsupervised Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*. [Online]. 14 (4). p.p. 47. Available from: https://doi.org/10.1145/3394053. [Accessed: 29 June 2021].

Mathur, P. & Burns, M.L. (2019). Artificial Intelligence in Critical Care.

International Anesthesiology Clinics. [Online]. 57 (2) p.pp. 89–102. Available from: https://journals.lww.com/anesthesiaclinics/Fulltext/2019/05720/Artificial_Intelligence_in_Critical_Care.8.aspx. [Accessed: 11 August 2021].

Mbaabu, O. (2021). *The basics of genetic algorithms in machine learning*. [Online]. 2021. Available from: https://www.section.io/engineering-education/the-basics-of-genetic-algorithms-in-ml/. [Accessed: 2 August 2021].

McCarthy, J., Minsky, M.L., Rochester, N. & Shannon, C.E. (2006). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*. [Online]. 27 (4). p.pp. 12–12. Available from: https://ojs.aaai.org/index.php/aimagazine/article/view/1904. [Accessed: 16 July 2021].

McQueen, R. & Knussen, C. (2002). *Research Methods for Social Science*. Prentice Hall.

Mitri, N., Marrouche, W., Awad, M. & Habib, R. (2017). Irregular breathing detection in CPAP assisted patients using hierarchical temporal memory. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. [Online]. November 2017, IEEE, pp. 1–6. Available from: http://ieeexplore.ieee.org/document/8285370/. [Accessed: 12 May 2018].

Moyano, A., Stępniak, M., Moya-Gómez, B. & García-Palomares, J.C. (2021). Traffic congestion and economic context: changes of spatiotemporal patterns of traffic travel times during crisis and post-crisis periods. *Transportation*. [Online]. 48 (6). p.pp. 3301–3324. Available from: https://link.springer.com/article/10.1007/s11116-021-10170-y. [Accessed: 1

October 2022].

Muthusenthil, B. & Kim, H. sung (2018). CCTV surveillance system, attacks and design goals. *International Journal of Electrical and Computer Engineering*. 8 (4). p.pp. 2072–2082.

Mutuvi, S. (2019). *Introduction to Machine Learning Model Evaluation*. [Online]. Available from: https://heartbeat.comet.ml/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f.

Al Najada, H. & Mahgoub, I. (2016). Big vehicular traffic data mining: Towards accident and congestion prevention. In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2016, pp. 256–261.

Das Nath, M. & Bhattasali, T. (2020). ANOMALY DETECTION USING MACHINE LEARNING APPROACHES. *Azerbaijan Journal of High Performance Computing*. [Online]. 3 (2). p.pp. 196–206. Available from: http://azjhpc.com//issue6/doi.org.10.32010.26166127.2020.3.2.196.206.pdf.

Neelakandan, S., Berlin, M.A., Tripathi, S., Devi, V.B., Bhardwaj, I. & Arulkumar, N. (2021). IoT-based traffic prediction and traffic signal control system for smart city. *Soft Computing*. [Online]. 25 (18). p.pp. 12241–12248. Available from: https://dl.acm.org/doi/10.1007/s00500-021-05896-x. [Accessed: 1 October 2022].

Nižetić, S., Šolić, P., López-de-Ipiña González-de-Artaza, D. & Patrono, L. (2020). Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of Cleaner Production*. 274.

Numenta (2014). *Geospatial Tracking Learning the Patterns in Movement and Detecting Anomalies Geospatial Tracking Executive Summary Advances in Anomaly Detection*. [Online]. 2014. Numenta, Inc. Available from: https://numenta.com/assets/pdf/whitepapers/Geospatial Tracking White Paper.pdf.

Numenta (2005). *Leading the New Era of*. [Online]. 2005. Available from: https://numenta.com/. [Accessed: 30 July 2017].

Ozbayoglu, M., Kucukayan, G. & Dogdu, E. (2016). A real-time autonomous highway accident detection model based on big data processing and computational intelligence. *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*. p.pp. 1807–1813.

Pan, J. & McElhannon, J. (2018). Future Edge Cloud and Edge Computing for Internet of Things Applications. *IEEE Internet of Things Journal*. 5 (1). p.pp. 439–449.

Passow, B.N., Elizondo, D., Chiclana, F., Witheridge, S. & Goodyer, E. (2013). Adapting traffic simulation for traffic management: A neural network approach. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. [Online]. October 2013, IEEE, pp. 1402–1407. Available from: http://ieeexplore.ieee.org/document/6728427/. [Accessed: 4 February 2018].

Pike, T.D. (2019). *Computational tools to support analysis and decision making*. [Online]. 11006. p.p. 110060O. Available from: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11006/110060O/Computational-tools-to-support-analysis-and-decision-making/10.1117/12.2518693.full. [Accessed: 11 August 2021].

Polson, N.G. & Sokolov, V.O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*. 79. p.pp. 1–17.

Powers, D.M.W. (2020). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. [Online]. (January 2011). Available from: http://arxiv.org/abs/2010.16061.

Price, R. (2011). Hierarchical Temporal Memory Cortical Learning Algorithm for Pattern Recognition on Multi-core Architectures. *Dissertations and Theses*. [Online]. Available from: https://pdxscholar.library.pdx.edu/open_access_etds/202. [Accessed: 25 July 2021].

Purdy, S. (2016a). Encoders. In: *Biological and Machine Intelligence*. [Online]. Available from: http://numenta.org. [Accessed: 21 January 2022].

Purdy, S. (2016b). *Encoding Data for HTM Systems*. [Online]. Available from: https://arxiv.org/abs/1602.05925v1. [Accessed: 24 July 2021].

Raj, J.S. (2019). A COMPREHENSIVE SURVEY ON THE COMPUTATIONAL INTELLIGENCE TECHNIQUES AND ITS APPLICATIONS. *Journal of ISMAC*. [Online]. 01 (03). p.pp. 147–159. Available from: http://irojournals.com/iroismac/. [Accessed: 11 August 2021].

Rajeswari, A.M., Yalini, S.K., Janani, R., Rajeswari, N. & Deisy, C. (2018). A Comparative Evaluation of Supervised and Unsupervised Methods for Detecting Outliers. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*. (Icicct). p.pp. 1068–1073.

Rani, A., Kumar, N., Kumar, J. & Sinha, N.K. (2022). Machine learning for soil moisture assessment. In: *Deep Learning for Sustainable Agriculture*. Elsevier, pp. 143–168.

Ray, S. (2017). *Commonly used Machine Learning Algorithms (with Python and R Codes)*. [Online]. 2017. Available from: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/. [Accessed: 2 August 2021].

Robson, C. & McCartan, K. (2016). *Real World Research*. 4th Ed. [Online]. Wiley-Blackwell. Available from: https://uwe-repository.worktribe.com/output/915824.

Samal, S.R., Mohanty, M. & Santhakumar, S.M. (2021). Adverse Effect of Congestion on Economy, Health and Environment Under Mixed Traffic Scenario. *Transportation in Developing Economies*. [Online]. 7 (2). p.p. 15. Available from: https://doi.org/10.1007/s40890-021-00125-4.

Sammut, C. & Webb, G.I. (2011). *Encyclopedia of machine learning*. Springer Science \& Business Media.

Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science 2021 2:3*. [Online]. 2 (3). p.pp. 1–21. Available from: https://link.springer.com/article/10.1007/s42979-021-00592-x. [Accessed: 13 August 2021].

Sarmah, D.K. (2020). A survey on the latest development of machine learning in genetic algorithm and particle swarm optimization. In: *Algorithms for Intelligent Systems*. Singapore: Springer Singapore, pp. 91–112.

Saunders, M., Lewis, P., Thornhill, A. & Bristow, A. (2019). *'Research Methods for Business Students' Chapter 4: Understanding research philosophy and approaches to theory development*. In: pp. 128–171.

Del Ser, J., Osaba, E., Sanchez-Medina, J., Fister, I. & Fister, I. (2020). Bioinspired Computational Intelligence and Transportation Systems: A Long Road Ahead. *IEEE Transactions on Intelligent Transportation Systems*. [Online]. 21 (2). p.pp. 466–495. Available from: https://ieeexplore.ieee.org/document/8661647/. [Accessed: 20 June 2022].

Shetty, P. & Singh, S. (2021). Hierarchical clustering: a survey. *International Journal of Applied Research*. 7 (4). p.pp. 178–181.

Simmons, M. (2014). *Traffic Congestion to Cost the UK Economy More Than £300 Billion Over the Next 16 Years - INRIX*. [Online]. London. Available from: https://inrix.com/press-releases/traffic-congestion-to-cost-the-uk-economy-more-than-300-billion-over-the-next-16-years/. [Accessed: 9 July 2021].

Sousa, R., Lima, T., Abelha, A. & Machado, J. (2021). Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting. *Electronics*. 10 (14). p.p. 1630.

Srinivasan, C.R., Rajesh, B., Saikalyan, P., Premsagar, K. & Yadav, E.S. (2019). A Review on the Different Types of Internet of Things (IoT). *Journal of Advanced Research in Dynamic and Control Systems*. [Online]. Volume 11 (Issue 1). p.pp. 154–158. Available from: http://www.jardcs.org/abstract.php?id=20. [Accessed: 12 August 2021].

Sturari, M., Catani, L., Mancini, A. & Frontoni, E. (2016). An integrated mobility system using real-time data for traffic simulation. *MESA 2016 - 12th*

IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - Conference Proceedings. (Dii).

Syed, A.S., Sierra-Sosa, D., Kumar, A. & Elmaghraby, A. (2022). Making Cities Smarter—Optimization Problems for the IoT Enabled Smart City Development: A Mapping of Applications, Objectives, Constraints. *Sensors*. [Online]. 22 (12). p.p. 4380. Available from: https://www.mdpi.com/1424-8220/22/12/4380. [Accessed: 17 June 2022].

Tamir, T.S., Xiong, G., Li, Z., Tao, H., Shen, Z., Hu, B. & Menkir, H.M. (2021). Traffic Congestion Prediction using Decision Tree, Logistic Regression and Neural Networks. *IFAC-PapersOnLine*. [Online]. 53 (5). p.pp. 512–517. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2405896321002627. [Accessed: 29 September 2022].

Tang, J., Liu, F., Zou, Y., Zhang, W. & Wang, Y. (2017). An Improved Fuzzy Neural Network for Traffic Speed Prediction Considering Periodic Characteristic. *IEEE Transactions on Intelligent Transportation Systems*. 18 (9). p.pp. 2340–2350.

Tang, J., Zou, Y., Ash, J., Zhang, S., Liu, F. & Wang, Y. (2016). Travel Time Estimation Using Freeway Point Detector Data Based on Evolving Fuzzy Neural Inference System. *PLoS ONE*. [Online]. 11 (2). p.p. e0147263. Available from: http://dx.doi.org/10.1371/journal.pone.0147263.

Tatbul, N., Lee, T.J., Zdonik, S., Alam, M. & Gottschlich, J. (2018). Precision and recall for time series. In: *Advances in Neural Information Processing Systems*. [Online]. 8 March 2018, Neural information processing systems foundation, pp. 1920–1930. Available from: http://arxiv.org/abs/1803.03639. [Accessed: 24

June 2021].

Tillmann, A.M. (2014). *On the Computational Intractability of Exact and Approximate Dictionary Learning*. [Online]. p.pp. 1–5. Available from: http://arxiv.org/abs/1405.6664.

Tisljaric, L., Majstorovic, Z., Erdelic, T. & Caric, T. (2020). Measure for traffic anomaly detection on the urban roads using speed transition matrices. In: *2020 43rd International Convention on Information, Communication and Electronic Technology, MIPRO 2020 - Proceedings*. 28 September 2020, Institute of Electrical and Electronics Engineers Inc., pp. 252–259.

Units, T.M. (2018). *National Traffic Information Service DATEX II Service*. [Online]. Available from: http://www.trafficengland.com/resources/cms-docs/overview.pdf.

Urbanowicz, R.J. & Moore, J.H. (2009). *Learning Classifier Systems : A Complete Introduction , Review , and Roadmap*. 2009.

De Villiers, M.R. (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*. 2005, pp. 142–151.

Vinuesa, R., Azizpour, H., Leite, I., Balaam, M., Dignum, V., Domisch, S., Felländer, A., Langhans, S.D., Tegmark, M. & Fuso Nerini, F. (2020). The role of artificial intelligence in achieving the Sustainable Development Goals. *Nature Communications*. 11 (1).

Walter, F., Sandner, M., Rcöhrbein, F. & Knoll, A. (2017). Towards a neuromorphic implementation of hierarchical temporal memory on SpiNNaker. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. May 2017, pp. 1–4.

Wang, L. (2019a). From Intelligence Science to Intelligent Manufacturing. *Engineering*. 5 (4) p.pp. 615–618.

Wang, P. (2019b). On Defining Artificial Intelligence. *Journal of Artificial General Intelligence*. 10 (2). p.pp. 1–37.

Wang, Y., Cao, J., Li, W. & Gu, Ts.W. (2017). Exploring traffic congestion correlation from multiple data sources. *Pervasive and Mobile Computing*. 41.

Wu, J., Zeng, W. & Yan, F. (2018). Hierarchical Temporal Memory method for time-series-based anomaly detection. *Neurocomputing*. [Online]. 273. p.pp. 535–546. Available from: https://www.sciencedirect.com/science/article/pii/S0925231217313887. [Accessed: 6 February 2018].

Wu, Y., Dai, H.-N., Wang, H., Xiong, Z. & Guo, S. (2022). A Survey of Intelligent Network Slicing Management for Industrial IoT: Integrated Approaches for Smart Transportation, Smart Energy, and Smart Factory. *IEEE Communications Surveys & Tutorials*. [Online]. 24 (2). p.pp. 1175–1211. Available from: https://ieeexplore.ieee.org/document/9732420/. [Accessed: 26 September 2022].

Xia, Z., Gong, J., Yu, H., Ren, W. & Wang, J. (2022). Research on Urban Traffic Incident Detection Based on Vehicle Cameras. *Future Internet*. [Online]. 14 (8). p.p. 227. Available from: https://www.mdpi.com/1999-5903/14/8/227.

Xuan, S., Kanasugi, H. & Shibasaki, R. (2016). DeepTransport: Prediction and simulation of human mobility and transportation mode at a citywide level. *IJCAI International Joint Conference on Artificial Intelligence*. 2016-Janua. p.pp. 2618–2624.

Yang, Q., Wang, J., Song, X., Kong, X., Xu, Z. & Zhang, B. (2015). Urban traffic congestion prediction using floating car trajectory data. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [Online]. 18 November 2015, Springer, Cham, pp. 18–30. Available from: https://link.springer.com/chapter/10.1007/978-3-319-27122-4_2. [Accessed: 15 July 2021].

Young, M., Varpio, L., Uijtdehaage, S. & Paradis, E. (2020). The Spectrum of Inductive and Deductive Research Approaches Using Quantitative and Qualitative Data. *Academic Medicine*. [Online]. 95 (7). p.p. 1122. Available from: https://journals.lww.com/academicmedicine/Fulltext/2020/07000/The_Spectrum_of_Inductive_and_Deductive_Research.41.aspx. [Accessed: 9 September 2022].

Yu, K.H., Beam, A.L. & Kohane, I.S. (2018). Artificial intelligence in healthcare. *Nature Biomedical Engineering*. [Online]. 2 (10). p.pp. 719–731. Available from: https://doi.org/10.1038/s41551-018-0305-z. [Accessed: 12 August 2021].

Yuan, Z. & Tu, C. (2017). Short-term traffic flow forecasting based on feature selection with mutual information. *AIP Conference Proceedings*. 1839 (May).

Yujun, C., Juhua, P., Jiahong, D., Yue, W. & Zhang, X. (2019). Spatial-temporal traffic outlier detection by coupling road level of service. *IET Intelligent*

Transport Systems. 13 (6). p.pp. 1016–1022.

Zaki, J.F.W. & Ali-Eldin, A.M.T. (2018). Towards effective traffic congestion prediction in Egypt. *2017 IEEE 8th International Conference on Intelligent Computing and Information Systems, ICICIS 2017*. 2018-Janua (Icicis). p.pp. 232–236.

Zantalis, F., Koulouras, G., Karabetsos, S. & Kandris, D. (2019). A review of machine learning and IoT in smart transportation. *Future Internet*. 11 (4). p.pp. 1–23.

Zhang, X., Zhang, J. & Zhong, J. (2018). Toward navigation ability for autonomous mobile robots with learning from demonstration paradigm. *International Journal of Advanced Robotic Systems*. [Online]. 15 (3). p.p. 172988141877793. Available from: http://journals.sagepub.com/doi/10.1177/1729881418777939.

Zhang, Z., Zhao, Y., Liao, X., Shi, W., Li, K., Zou, Q. & Peng, S. (2019). Deep learning in omics: A survey and guideline. *Briefings in Functional Genomics*. [Online]. 18 (1). p.pp. 41–57. Available from: https://academic.oup.com/bfg/article/18/1/41/5107348. [Accessed: 21 September 2022].

Zyarah, A.M. (2015). *Design and analysis of a reconfigurable hierarchical temporal memory architecture*. Rochester Institute of Technology.

Appendix A The Baseline Experiments

In this section, before designing and implementing the Multi-level Anomaly Detection Framework several experimental tests on the HTM model were conducted in order to utilizes a HTM algorithm and its components. These tests were the baseline for the actual framework.

The base-line experiments aim to answer the following questions:

1. Are the datasets in hand suitable and well prepared to be used?
2. Is HTM suitable for smart transport traffic management applications?
3. How to identify the best available encoder from those proposed by Numenta?

To answer the first question, as a secondary contribution in this thesis, the datasets were reviewed in Chapter 4. Furthermore, based on the most recent publications used in Table 4-1, it emerged that the proposed work needs a set of data that has the following features:

- A public domain dataset.
- A set of data features found from Table 4-1, which includes speed, flow, and time of the day features to be used to identify the level of congestion, which can be used as a predictive filed.

To answer the second and third questions, the researcher studied the HTM and its CLA implementation in Chapter 3. As a results, the rationale behind choosing HTM to detect and predict congestions as part of smart transport application domain was discussed in Section 3.7. Moreover, HTM can fit any application that uses streams of data (Hawkins & Ahmad, 2016; Wu et al., 2018). This applies well to transport applications. More specifically, transport applications deal with huge amounts of data streams that are constantly changing during the day whenever new events occur on roads. Such data streams require a self-learning algorithm with properties that enable it to adapt to these changes and at the same time save the system memory. These properties can be guaranteed with the use of HTM algorithm.

In the baseline experiments, the researcher used a model that is based on Cui, Ahmad and Hawkins' (2017) components of an end-to-end HTM system. It is an anomaly detection model. As shown in Figure A-1, the model particularly describes the workflow for the standard CLA components (encoder, SP, TM) which are discussed previously in detailed in Chapter 3. The results of the baseline experiments can tell us if the algorithm could fit a smart transport application.

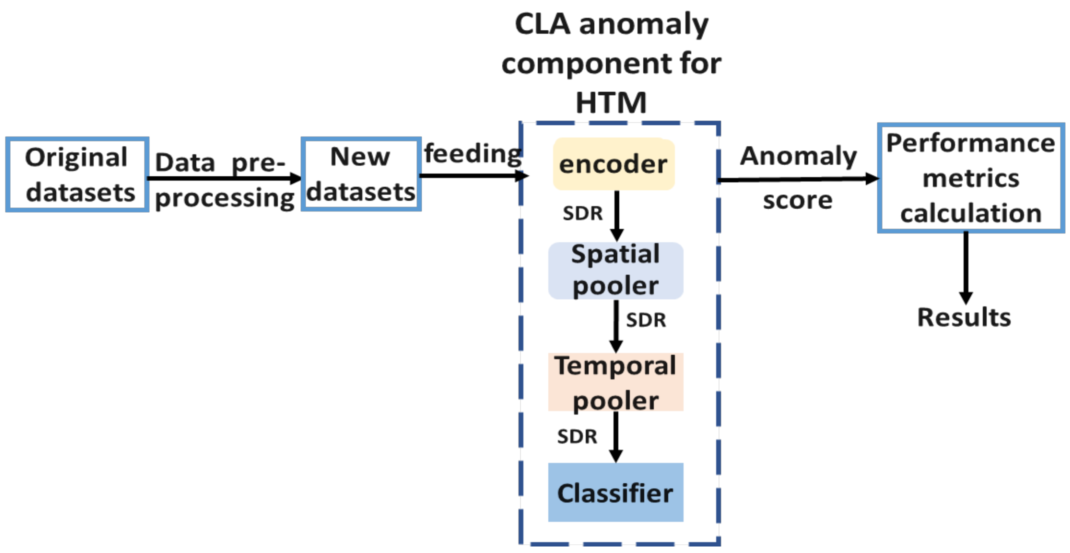
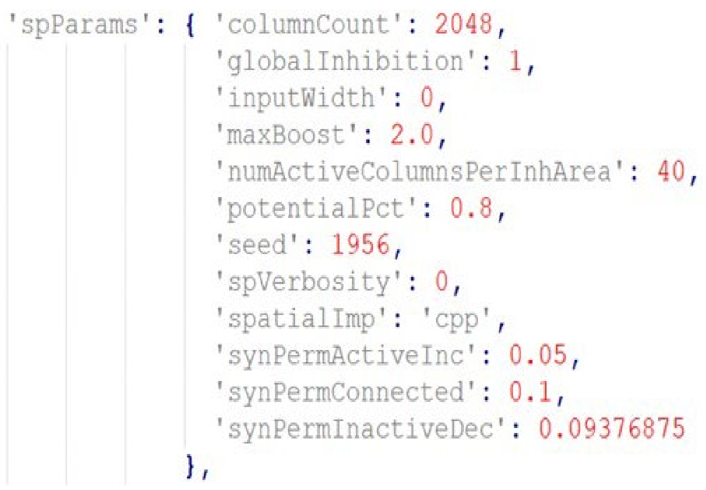


Figure A-1 :The flowchart of the anomaly detection CLA workflow from

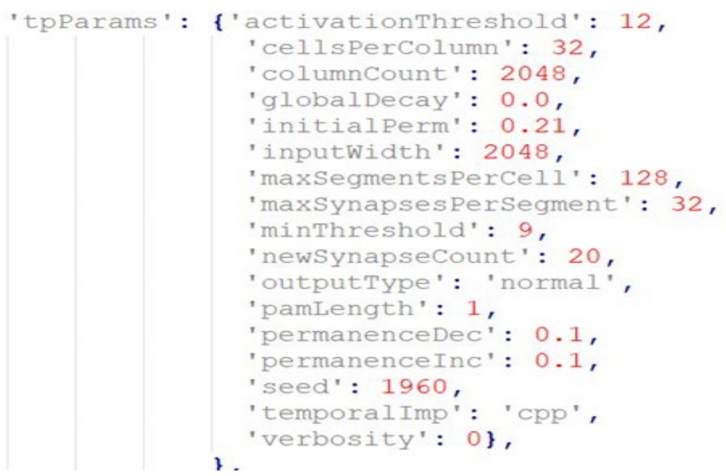
In the case of the third question, the researcher decided to use a Numenta open-source project NuPIC scalar encoder, and date encoder. Scalar encodes deal with numerical data types. They can be utilised to encode integers and floating-point numbers. The date encoder is similar to a scalar encoder, but it provides a better interface for dealing with dates.

This section previews the stage of the project where the pre-processed datasets are fed into the slandered CLA algorithm to produce anomaly scores. Two experiments are conducted on the same model parameters for anomaly detection HTM cortical learning algorithm. In these experiments, there are no differences between the spatial pooler and temporal pooler parameter settings, as the HTM model is built using parameter settings recommended by NuPIC. Figure A-2 and Figure A-3 show the parameter settings for the spatial pooler and temporal memory algorithms selected from NuPIC.



```
'spParams': { 'columnCount': 2048,
              'globalInhibition': 1,
              'inputWidth': 0,
              'maxBoost': 2.0,
              'numActiveColumnsPerInhArea': 40,
              'potentialPct': 0.8,
              'seed': 1956,
              'spVerbosity': 0,
              'spatialImp': 'cpp',
              'synPermActiveInc': 0.05,
              'synPermConnected': 0.1,
              'synPermInactiveDec': 0.09376875
            },
```

Figure A-2: Spatial pooler parameters settings recommended by NuPIC.



```
'tpParams': {'activationThreshold': 12,
             'cellsPerColumn': 32,
             'columnCount': 2048,
             'globalDecay': 0.0,
             'initialPerm': 0.21,
             'inputWidth': 2048,
             'maxSegmentsPerCell': 128,
             'maxSynapsesPerSegment': 32,
             'minThreshold': 9,
             'newSynapseCount': 20,
             'outputType': 'normal',
             'pamLength': 1,
             'permanenceDec': 0.1,
             'permanenceInc': 0.1,
             'seed': 1960,
             'temporalImp': 'cpp',
             'verbosity': 0},
},
```

Figure A-3: Temporal memory parameters settings recommended by NuPIC.

To identify the best encoder, they would give a better CLA performance, two different encoders, namely, the scalar and the coordinate encoders are used. The proposed workflow is charted in Figure A-1 . First, the algorithm is trained using 80% of the data that represent a normal traffic, that is, where there is no congestion. Then, the rest of the data which represent both normal and abnormal traffic is fed to the trained algorithm to first detect any anomalies and then, if found, to assign them a score. The results from this stage are then evaluated against several states of the art anomaly detecting algorithms using the latest version of RapidMiner platform, by computing performance. The experiments are run via the Numenta Platform for Intelligent Computing (NuPIC) framework. NuPIC is an open source from Numenta that implements the HTM codebase, which has been structured to run algorithms in Python (Numenta, 2014b). In this part of the experiments, the proposed model first uses the scalar encoder to detect traffic anomalies. The model scores between 0 and 1. A value of 0 would be normal; any other value would be abnormal and would automatically be assigned to an anomaly score field. Following the identification of the anomaly scores, the evaluation platform RapidMiner studio is used to calculate the Precision, Recall and F-measure as explained in section 5.4.3. The settings of this experiment are listed in Table A-1.

Table A-1: Settings for baseline experiments

Encoder	**Coordinate/ Scalar**
Dataset Format	CSV
Input feature	Speed, congestion label, Traffic flow
output	Anomaly score
Total rows for training	3224
Total rows for testing	807
platform	NuPIC version1.0.5
Programming language	Python 2.7.3.
Operating System	Linux Ubuntu 7
Hardware	Processor: Intel(R) Core(TM) i7 CPU @ 2.30 GHz RAM:12.0 GB System type: 64-bit.
Evaluation platform	RapidMiner version 9.3

To get the highest possible accuracy performance, the obtained anomaly scores with a threshold that varies between 0.1 and 0.9 were compared to find the best threshold performance using the scalar and coordinate encoders. The results are plotted in Figure A-4, Figure A-5, Figure A-6, and Figure A-7 for the four performance evaluation metrics. In all of the charts below, and for every threshold point, the model performs consistently better with the scalar encoder than with the coordinate encoder.

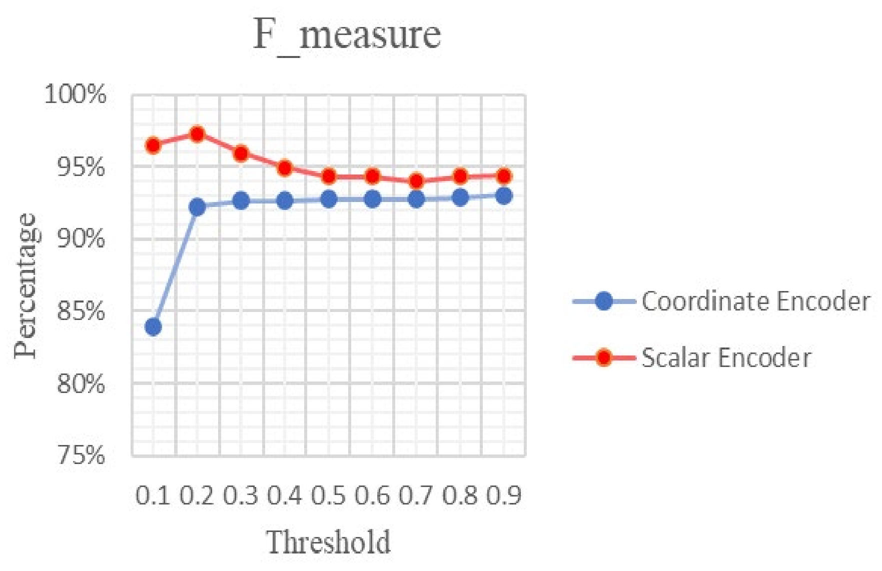


Figure A-4: F-mrasure comparison.

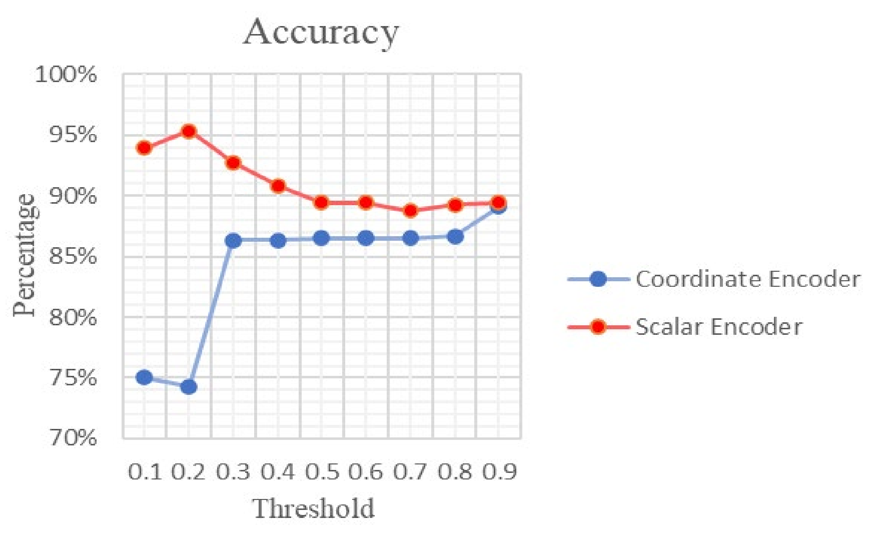


Figure A-5: Accuracy comparison.

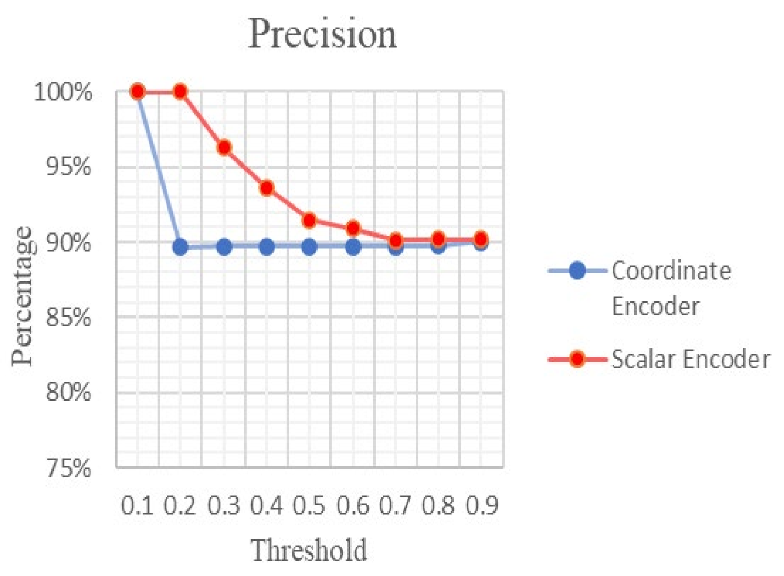


Figure A-7: Precision comparison.

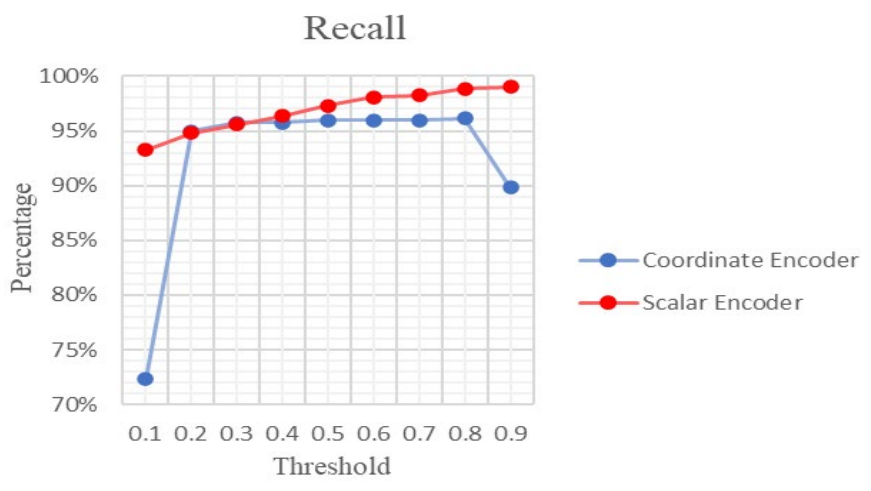


Figure A-6: Recall comparison.

Along the same lines, the Precision values for the model with the scalar encoder are either very close to or higher than the values for the coordinate encoder, never lower than them. Another point worth noting, here, is that there is a slight drop in the trajectory of the model with the scalar encoder from threshold points 0.3 to 0.6, as illustrated in Figure A-6. Finally, Recall outcome does not deviate from the overall pattern of the difference in the performance of the two encoders. Here, again, the Recall scores of the model with the scalar encoder are higher than with the coordinate encoder for all the threshold points except for points 0.2 and 0.3, where both encoders perform almost the same. Figure A-7 charts the Recall performance of the two encoders.

The results collectively indicate that the scalar encoder model is more suitable for the proposed algorithm. The results also show that the threshold point where the model generally performed the best is 0.2. In summary, the scalar encoder model shows more promising results that can be manipulated to get more accurate results with different complicated scenarios.

Regarding the contribution of the encoder to the performance of the proposed algorithm, it is clear from the figures that the model with the scalar encoder outperforms the model with the coordinate encoder along all four metrics. The accuracy of the model with the scalar encoder scores approximately 95.32%, nearly 4.52% higher than the accuracy of the model with the coordinate encoder. Precision-wise, the difference between the encoders on HTM seems negligible (0.08%), but the directionality of the difference is still in favour of the scalar encoder. The Recall metric is also higher for the model with the scalar encoder by 2%. Expectedly, the F-measure is higher for the model with the scalar encoder than it is for the model with the coordinate encoder by about 1.5%.

These results confirm both the suitability and potential of HTM to be harnessed for traffic detection data in general, and for anomaly detection, more specifically. The next step is to work on the HTM model with large datasets that comprise of more complicated rea-life road traffic scenarios. It must be remembered that the baseline experiments reported in this section are limited by the fact that they have not taken time into account. Traffic detection and management are naturally impacted by time dynamics in numerous ways. So, in the next stage, the temporal parameters of Date and time stamps will be included. Another potentially limiting factor is that the baseline experiments have used the NuPIC default settings.

The baseline results confirm both the suitability and potential of HTM to be harnessed for traffic detection data in general, and for anomaly detection, more specifically. The next step is to work on the HTM model with large datasets that comprise of more complicated rea-life road traffic scenarios. It must be remembered that the baseline experiments reported in this section are limited by the fact that they have not taken time into account. Traffic detection and management are naturally impacted by time dynamics in numerous ways. So, in the next stage, the temporal parameters of Date and time stamps will be included. Another potentially limiting factor is that the baseline experiments have used the NuPIC default settings. In the next stage, more settings will be manipulated and tested.