FLINDERS
UNIVERSITY

ADELAIDE
AUSTRALIA

Online Learning for Short Term Arterial Traffic Prediction and Incident Detection

by

Jonathan Glenn Mackenzie, *B.Eng.(Software) (Hons)*
College of Science and Engineering

July 24, 2020

A thesis presented to
Flinders University
in total fulfilment of the requirements for the degree of
Doctor of Philosophy

Adelaide, South Australia, 2020

Contents

List of Figures

List of Tables

Abstract

Sydney Coordinated Adaptive Traffic System (SCATS) is an Intelligent Transportation System currently deployed in Adelaide, Australia, that is responsible for (amongst other tasks) controlling the sequences, cycles and timing at signalised road intersections, and collecting and storing a record of these operations and vehicle flow counts. Within the context of this SCATS data, the research presented in this thesis investigates:

- How traffic flows at specific intersections can be used to make predictions about future traffic volumes in the short term, specifically, in the next phase of traffic and on aggregate over the next 5/10/15 minutes.
- Whether outliers within these observations and predictions can indicate that traffic flow is indicative of an incident or otherwise anomalous traffic behaviour.

The task of answering these questions is of *global* significance, as the effectiveness of the tools that solve these problems has the potential to impact the safety and efficiency of travel for hundreds of millions of commuters every single day, as they travel through road networks monitored by ITS.

In this work, traffic data for the Adelaide metropolitan arterial road network are used to evaluate a variety of existing and novel models on their predictive performance. The dataset has two distinct road areas:

Central Business District: those monitored intersections within the CBD. Traffic flow in this area is not typical of that on arterial areas due to its unique traffic usage among other factors.

Arterial: those high flow intersections that are outside the CBD but are not freeways.

HTM, LSTM, ARIMA and Markov models are predictive evaluated for short term arterial traffic prediction in 5 minute aggregate and on a next-phase basis. HTM and LSTM models perform particularly well at both tasks. The practical implications of this finding allows for the development of traffic control systems that act in a proactive (rather than a reactive manner).

HTM is also evaluated for its ability to indicate the presence of an incident purely from prediction anomalies and compared to SHESD (a statistical time

series anomaly detection algorithm). While anomalous data exists in the SCATS dataset, these anomalies do not imply the presence of an incident and incidents do not necessarily cause anomalous vehicle flow counts. Practically, this means that anomalies in vehicle flows should not be used to infer the presence of an incident.

Certification

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

As requested under Clause 14 of Appendix D of the *Flinders University Research Higher Degree Student Information Manual* I hereby agree to waive the conditions referred to in Clause 13(b) and (c), and thus

- Flinders University may lend this thesis to other institutions or individuals for the purpose of scholarly research;
- Flinders University may reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Jonathan Glenn Mackenzie
November 2019

Acknowledgements

I would like to thank:

- My partner, Laura, and son, Logan, for putting up with me.
- My supervisors, Prof. John F. Roddick, Prof. Rocco Zito and Dr. Carl Mooney.
- Flinders University for providing me with the FURS scholarship, the space, and resources to do my research.
- DPTI for providing SCATS and incident data for the Adelaide area.
- Dr. Denise de Vries and Dr. Anna Shillabeer for providing me with valuable advice and feedback.
- Prof. Rocco Zito, Branko Stazic and Tai Dinh for providing me with the traffic datasets and a wealth of knowledge about the data, associated systems and their operation.
- Dr. Darfiana Nur for helping with my understanding and use of SARIMA models.

Jonathan Glenn Mackenzie
jonathan.mackenzie@flinders.edu.au
July 24, 2020
Adelaide, Australia

Chapter 1

Introduction

1.1 Intelligent Transportation Systems

Intelligent transportation systems (ITS) utilise a wide range of technologies that automatically control transportation infrastructure with the aim of reducing inefficiencies in large and complex systems. Improvements in factors such as travel time, congestion, health and safety are paramount when designing ITS applications, given that road transport is an indispensable component of modern life and the average Australian commuter spends at least 1 hour on the road every day (Core Data 2016).

In this chapter, ITS will be introduced in broad terms and within the specific context of the research in this thesis, research questions described and .

1.2 Motivation

There is a significant need for such systems now and into the future given that:

- The population of humans is continuously growing, projected to reach 9.7 billion by 2050 (United Nations 2017) who will require persons and objects to be transported between physical locations.
- The majority of this transport will occur on a road. In 2017, in Australia there were 300.7 billion passenger kilometres and 213.9 billion freight kilometres travelled on roads (BITRE 2017). These numbers far outstrip the other modes of transport: rail, sea and air and are only projected to increase according to BITRE (2017), see Figure 1.1.
- According to the OECD, as the wealth of the world's population increases, so too will their spending power and desire for transportation and transported goods (Kharas 2010). For example in China there were over 300

million registered vehicles in 2018 (Zheng 2018) (up from 250 million in 2015 (Toroyan 2015)) and a population of 1.3 billion with an increasing average household wealth profile (Stratford & Cowling 2016).

- In 2018, the World Health Organisation (WHO) reported that road traffic injuries are the eighth leading cause of death across all age groups worldwide and that there is a road related death every 24 seconds accounting for 1.35 million deaths annually (Toroyan 2018). In the USA, 94% of traffic related fatalities are caused by human error (Terry & Tanner 2018). ITS can play a significant role in increasing safety and reducing road mortality.

- Santos (2017) urges the need to reduce greenhouse gas (GHG) emissions to keep global temperature rise below 2°C and prevent a catastrophic global warming event, which necessitates the optimisation of all infrastructure in terms of emissions, of which road transportation accounted for 20% globally in 2017. ITS is a key technology in the optimisation of road transport flow and, the work presented in this thesis evidences an opportunity for a big-data driven approach to traffic control systems and thus a reduction in GHG emissions.

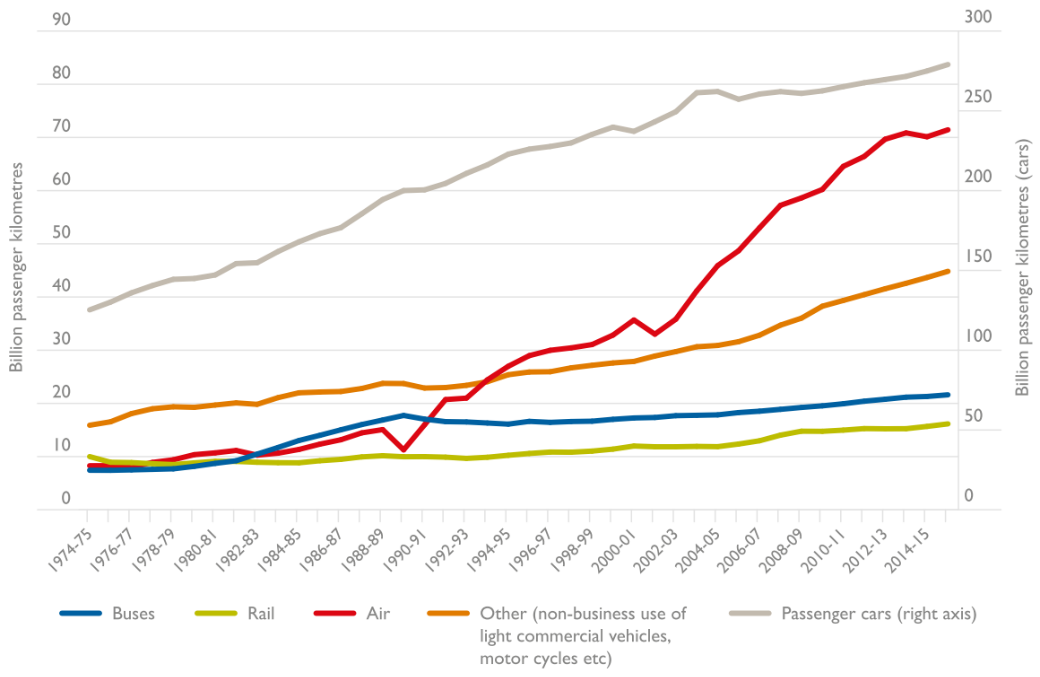


Figure 1.1: Estimates of Australian Domestic Passenger Travel by Mode BITRE (2017)

The WHO reports that low-income nations, with only 1% of vehicles, account for 13% of all deaths, whereas 40% of vehicles in high-income nations account for 7% of deaths. It is clear here that the deployment of ITS plays a key role in saving lives, especially in developing nations where the cost of deploying ITS is far cheaper than constructing additional roads. Khanal (2012) posits that, given the limited space and financial constraints in countries such as India, ITS provides a way to ease existing congestion and related issues through more effective traffic control. For example:

- Incident detection can be used to deploy emergency services to incident locations and rerouting traffic, easing resultant congestion and reducing the likelihood of subsequent incidents.
- The increased level of control over traffic flow may also alleviate dangers posed by non-existent or otherwise uncoordinated traffic control.
- Data collected and analysed by these systems can be used to better inform the control of signals and the design of new infrastructure based on observed and forecast flows.

Considering these applications of ITS to road transport efficiency, safety, GHG emissions, research into intelligent systems to control and manage these growing transport networks is essential. In light of this, research in this thesis is focused on 2 main areas within the context of arterial traffic networks: the short term prediction of traffic flow and the detection of incidents from raw traffic flow data. Research into solutions to the first issue will allow for improved traffic signal timings which account for expected flow, and the second will improve emergency response and incident clearing time, both of which will address the above concerns.

1.3 Function

At a high level, ITS seeks to improve the efficiency and safety of transport systems via the collection and utilisation of data about the state of the system. Sussman (2008) describes the 6 main components of ITS as:

1. Advanced Transportation Management Systems (ATMS): systems that manage roadway functions based on real time collected data, such as congestion prediction and detection, providing alternate routing instructions to vehicles and maintaining priority for high-occupancy or emergency vehicles.
2. Advanced Traveller Information Systems (ATIS): systems that provide data directly to travellers, for things such as incident locations, weather issues, road conditions, road restrictions and optimal routes. This information may be communicated via mobile phone or internet applications directly accessible by road users.

3. Advanced Vehicle Control Systems (AVCS): systems that provide enhanced vehicle control to improve safety and efficiency. Such features include collision detection and avoidance, semi-autonomous or fully autonomous driving. This functionality is made possible by the fusion of various onboard sensors including: camera (including stereoscopic and traditional cameras), high accuracy global positioning systems (GPS), radio detection and ranging (RADAR), light imaging detection and ranging (LIDAR), proximity sensors, powerful onboard computation facilities and advanced models that control the vehicle. Safety features include driver fatigue detection and alarms (Clement et al. 2015), and lane departure warning (Suzuki & Jansson 2003).

   These features may also integrate between vehicles and other infrastructure (including devices on the roadside, embedded in the road, or at greater distances via radio communication) to further augment their operation. Improvements in this area go beyond passenger safety and will see greater transport accessibility for seniors, the disabled and those who would otherwise have limited mobility.

4. Business Vehicle Management Systems (BVMS): these are systems that allow businesses to track and monitor their fleet vehicles in car, truck, taxi and bus domains. Such tools allow fleet owners and managers to track the fuel consumption, driver behaviours (such as poor driving, rule violations and downtime), vehicle maintenance status and vehicle life cycle management.

5. Advanced Public Transportation Systems (APTS): systems that increase the public accessibility of information about the state of public transport such as arrival times, delays, service substitutions and route changes, as well as to provide operators with current information for monitoring and planning.

6. Advanced Rural Transportation Systems: systems that address the special constraints of relatively low-density, high speed roads. Such road networks can only be sparsely monitored and serviced due to the relatively large distances involved between road users and services. There is very little literature on these types of specialised systems, indicating future research opportunities, especially as the deployment of global telecommunications services expands to provide coverage in remote areas.

The full potential of these components has not been fully realised or implemented yet. For example, fully autonomous vehicles have yet to overcome significant technological, legal, social and ethical hurdles:

- Bonnefon et al. (2016) report the results of surveys where people would paradoxically approve of and prefer that other people buy and use autonomous vehicles that take a utilitarian action in making a decision between injuring small number of passengers over a larger number of bystanders, but would not buy or ride in such a vehicle themselves.

- Howard & Dai (2014) report that public perception of an acceptable injury rate is far higher than that for human drivers. Other barriers to adoption were cost and perceived levels of control, liability and safety.
- Terry & Tanner (2018) report the numerous road conditions in which autonomous vehicles experience significant difficulty in navigating safely. For example, autonomous vehicles systems that rely heavily on video footage in the visible light spectrum do not handle heavy snow safely.
- Legal and regulatory issues of control and liability are only recently being considered in Australia, where laws are being amended in order to allow for limited and tightly controlled trials while in the USA, regulations in a number of states allow for autonomous vehicle operation that place accident liability on the manufacturer with far greater insurance requirements (Goplan 2018). Goldstein (2016) reports that governments need to find a balance in legislation that does not prohibit the development of such technology, but still maintains an acceptable level of safety and just legal outcomes in case of accidents. For example, laws that require a human driver to have their hands on the wheel while driving in order to not be considered "reckless driving" eliminates many large benefits of autonomous vehicles.

1.4 Advanced Transportation Management Systems

The research in this thesis is chiefly concerned with ATMS, specifically the development of methods and systems to better predict traffic short term flows, and detect incidents from such data. In general terms, ATMS performs the following functions:

- Collection and monitoring of data about the state of infrastructure. This includes information such as:
  - Vehicle volume: the approximate number of vehicles on a particular stretch of road at any given time.
  - Degree of congestion: as road networks have a physical limit to the amount of vehicles that can travel upon them, this data reports the degree to which this capacity is reached or exceeded.

  - Link journey time (LJT): the average time it takes a vehicle to travel a given stretch of road during a particular time period.
  - Route journey time (RJT): the average time taken for a vehicle to travel between selected locations, and can be derived by combining several link journey times. Typically RJT systems (Cox 2014, 2013, Zhou et al. 2013, Michau et al. 2014, Thogulava et al. 2015) use Bluetooth detectors that monitor time between the detection of unique devices (usually in the form in-car hands-free mobile communication facilities or mobile phones themselves) throughout the road network.
  - Current state of traffic signals and vehicle detectors. This includes factors such as signal state, sensor occupancy and failure.

- Real-time adjustment of signal timings to optimise vehicle throughput. In the context of SCATS which is a proprietary product, there is no publicly available information on how this is achieved or by what mechanism.
- Adjustment of other signage to reroute traffic and inform network users of factors such as accidents and approximate travel times.
- Detection of incidents.
- Providing real time video feed to traffic monitoring staff.
- Informing system operators of hardware failures.

Once these factors are optimised, Dimitrakopoulos & Demestichas (2010) cites the secondary effects on a road-network as:

- Reduced travel time, as automatically adjusted signal controls and traffic routing ensure drivers take improved routes and intersections provide optimal timings according to current demand.
- Faster maintenance response times as infrastructure self-reports hardware degradation or failures. With more responsive maintenance, equipment downtime is reduced, the system is kept fully operational and drivers aren't required to navigate dangerous uncontrolled roads and intersections.
- Improved incident response time as incidents are detected rapidly and emergency services or other appropriate responders are deployed. With this, resultant congestion and risk of secondary incidents is eliminated. In the case of human injury, the mortality of victims is reduced as precious time is saved transporting them to hospitals.
- Improved traveller safety as signal control ensures that drivers and signals are acting in a predictable and orderly manner. The safety implications of ATMS are readily apparent when considering the incident rates in cities with ITS are far lower than those without (Toroyan 2018).

- Increased traveller satisfaction and improved quality of life as people see road travel as an acceptably safe and convenient mode of transport that allows them to achieve their goals in a timely manner without undue delays.
- Reduced non-renewable fossil fuel consumption leading to improved air quality as vehicles spend less time on the road, requiring less fuel to power them. This result is particularly pertinent given the ongoing negative impacts of climate change caused in part by greenhouse gas emissions generated road vehicles.

1.4.1 Research Issues in ATMS

There are numerous problems and research opportunities within ATMS. Due to the large number of deployed traffic management systems across the globe, developed by both governments and private enterprises, covering motorway, freeway, arterial road contexts, it comes as no surprise that there are numerous approaches to managing traffic. With no clear optimal strategy or standard traffic control method (evidenced by the ongoing issues of congestion and traffic jams), it is of vital importance that research into novel ATMS be actively engaged in concert with research into infrastructure construction, planning, and management to ensure safer and more efficient transport for all.

The upside to this fact means is that there are a wealth of data being generated and collected regarding ATMS, presenting numerous directions for research. Examining the direct outputs of these systems for tasks such as signal control, and the topics discussed in the present work: traffic flow prediction (see Chapter 4), and incident detection (see Chapter 6). These tasks are far from being solved problems, for example, the task of predicting traffic flow is an active area of research, Alsrehin et al. (2019) describes 165 different approaches to traffic flow prediction using machine learning or data mining techniques.

The enhancement of existing Traffic Management Systems (TMSs) and the development of new traffic management systems is of significant concern (de Souza, Brennand, Yokoyama, Donato, Madeira & Villas 2017, Nellore & Hancke 2016, Gettman et al. n.d., Coconea & Bellini 2019, de Souza, da Fonseca & Villas 2017), as new data sources become available (Pack et al. n.d.), opportunities arise to better manage existing traffic, especially in a world where semi-autonomous and fully autonomous vehicles become the norm and traffic may be managed at the level of individual or cohorts of vehicles. Whereas conventional ATMS simply uses existing sensors to monitor and control traffic flow, future systems may exploit auxiliary data sources such as social media (Chen et al. 2017, Fu et al. 2015), weather data, emergency data, satellite imagery, Internet-of-Things (IoT) devices (Lakshminarasimhan 2016), and in-vehicle sensors.

The so-called Internet-of-Things consists of self-configuring devices, or things, that connect to the internet and form the basis of smart cities, where once isolated devices such as parking meters or street lamps can communicate amongst themselves intelligently in order to deliver enhanced infrastructure experiences and management. Devi & Rukmini (2016) describe the development of a subset of the Internet-of-Things referred to as the Internet-of-Vehicles (IoV), in which connected vehicles (either in small transient clusters, in larger fleets or more globally) communicate their status between each other to coordinate their operation. Research issues abound, as such systems require investigation into their design, integration, privacy, infrastructure, networking, security, redundancy, and multiple levels of interoperability amongst heterogeneous elements. Brandl (2016) posits that the development of such systems may even lead the abandonment of stationary sensors as part of ATMS altogether.

The integration and communication between existing ITS such as ATMS, ATIS and APTS, where data flows enhance the operation of both systems (*Smarter and more connected: Future intelligent transportation system* 2018). For example, data collected by the ATMS can be communicated to travellers in order to inform their routes or status of public transport.

The realisation of such technology requires significant efforts in data standardisation to ensure that vehicles and ATMSs can communicate and coordinate to provide positive outcomes for commuters. This applies to communication between the various kinds of ITS, third party data sources utilised by ITS, and the communication of varying devices within the specific systems (vehicle-to-vehicle (V2V) and vehicle-to-anything (V2X) communication for example).

The mere existence of such large datasets and dependencies presents significant issues of security, resilience, and privacy (Zhang et al. 2020, de Souza, Brennand, Yokoyama, Donato, Madeira & Villas 2017). It is of critical importance that cyber-attacks cannot disrupt the function of ATMS that heavily rely on third-party data, and as such there are opportunities for research into existing and potential (novel or otherwise) attack vectors, and their mitigation. In the case of data-breaches or data-leakages, the anonymity of commuters must be maintained to ensure that personally identifying information cannot be recovered (Dukkipati et al. 2018).

1.4.2 Challenges Posed by ATMS Data

Considering the following facts about a typical city managed by a single ITS:

- There are hundreds and potentially thousands of signalised intersections.
- Each intersection (and selected links between intersections) have a number sensors and other inputs including:
  - Loop based vehicle detectors at stop lines
  - Vehicle detecting cameras

  - Pedestrian crossing buttons
  - Bicycle crossing buttons and detectors
  - Bus detectors

- These permitted movements through an intersection may cycle through multiple phases every minute.
- Stretches of road between two intersections, called links, may have additional loop detectors covering:
  - Mid block: to detect the presence of vehicles between intersections
  - Queuing detectors: to detect vehicles waiting at a road-level rail crossings
  - Incident detectors: on freeways, detectors are used to count and detect missing vehicles and the incidents that their absence may imply.
- It would not be economical to install and monitor closed circuit television beyond covering selected critical locations as additional sensors incur additional installation, monitoring, maintenance, data processing and storage costs.

The tasks of predicting traffic flow in the short term and detecting incidents within this network is far beyond that of human capabilities given the large amount of data being produced and its spatiotemporally varying nature. One advantage of such large amounts of data being produced by the current ITS in Adelaide, Australia is that hardware and algorithms now exist that can:

- Rapidly store and retrieve large amounts of data
- Use these large amounts of data to produce models that can effectively and efficiently:
  - Predict future data from and within the context of previously observed data where the distribution of that data changes over time. Such models are said to engage in "online learning".
  - Detect anomalies within data where that data has a seasonal component. That is, where the data has recurring patterns over time where anomalies may appear at different locations within those patterns.
  - Determine the ideal parameters of such models to ensure the highest accuracy possible.

Motivated by these challenges and opportunities, the research presented in this thesis focuses on two areas in an arterial context: short term traffic flow prediction (in the next phase of a intersections cycle and in aggregate) and automated detection of arterial incidents. Discussion of the development of a dataset containing traffic network and flow information is covered in section 2.5. Greater detail of these tasks and literature reviews of incident detection is covered in Chapter 6 and traffic flow prediction in Chapter 4.

1.5 Research Impact

The four main contributions described in this thesis are:

1. The creation of two new large datasets for researching traffic flows within the Adelaide metropolitan area:
   (a) Traffic signal location data describing the logical connections between signalised intersections used by SCATS in Adelaide
   (b) Traffic flow data, describing traffic flows through all intersections in Adelaide over a multi-year period
2. Demonstration of the use of HTM for predicting aggregated traffic flows with competitive results when compared to state of the art methods. These results were published in *IEEE Transactions on Intelligent Transportation Systems*, see Mackenzie et al. (2018). These efficacy of these models and a novel Markov based model are also demonstrated on a next phase traffic flow prediction task.
3. The development of a novel Markov based method for short term traffic flow prediction, with performance scores higher than those of other more complex methods.
4. The finding that anomalous traffic flows do not necessarily indicate accidents, and that accidents do not necessarily cause anomalous traffic flows.

These findings are accompanied by scripts and other software that readily allow others to validate the above results, or extend and use them for their own research purposes as described in section A.3.

1.6 Thesis Organisation

This thesis is organised as follows:

Chapter 1 Introduction to this thesis and ITS.

Chapter 2 Description and background into the datasets used in the research, their preprocessing and storage.

Chapter 3 Description of the Hierarchical Temporal Memory (HTM) algorithms used throughout the research.

Chapter 4 Report of research into the suitability of HTM and Long Short Term Memory (LSTM) for 5, 10 and 15 minute arterial traffic flow prediction.

Chapter 5 Report of research into suitability of HTM, LSTM, and Markov Models for 5 minute and next phase traffic prediction.

Chapter 6 Report of research into the use of HTM, and Seasonal Hybrid Extreme Studentized Deviate Test (SHESD) for arterial incident detection.

Chapter 7 Conclusion and recommendations for future work.

Appendices

Bibliography

Separate literature reviews are provided in chapters where appropriate. The author acknowledges the discrepancy between this organisation, and the standard thesis organisation of a distinct literature review chapter, but feels that the chosen format enhances the flow of the document, as literature relevant to each research area is kept in close proximity to the research itself.

1.6.1 Note About Notation

In this work, the following conventions (which are close to the semantics of the python programming language) are used in pseudocode:

- A list is a data structure that can store an arbitrary number of values in a sequence. Values are added to the end of a list L with $L.append(v)$.
- The number of elements in a list is retrieved via $L.length$ or $length(L)$.
- Retrieving a value at index i in an a list L is shown as $L[i]$. Negative indexes count from the end of the list, for example $L[-1]$ returns the last element, $L[-2]$ returns the 2^{nd} to last element.
- Retrieving a slice of a list, that is the sublist of elements from index i (inclusive, starting from 0) to index j (exclusive) (these indexes may be positive or negative as described above), is shown as $L[i : j]$. For example, if $L = [1, 2, 3, 4, 5]$, $L[2 : 4] = [3, 4]$.

- A map M is a data structure that associates a key k (a key can be any value such as integer, character string or a list of values) with a value v where values may be any values including lists or other maps. Associations are assigned using $M[k] \leftarrow v$. A value is retrieved using $M[k]$. If k is not in M, 'null' is returned.

- The keys and values of map are retrieved as a list by $M.keys$ and $M.values$ respectively.

- A counter C is a map data structure that counts occurrences of values. The default retrieved value for any key is 0. Counts for a particular key are incremented by $C[k]+=1$.

Dates described in this document use the least to most significant Australian date format: `DD/MM/YYYY`, for example 7/9/2018 is the 7th of September 2018. The exception to this is when describing dates in JSON data, in which case *ISO 8601* format is used: `YYYY-MM-DD HH:mm:ss.0000` (year, month, day, hour minute, second, milliseconds).

Chapter 2

Data Sources

A number of different data sources are used in this thesis, their collection, properties, extraction, storage and issues, are described in this chapter. The data is the basis for all models and their evaluation in this research. The datasets created are spatiotemporal in nature, that is, each record has an associated time and geographical location (amongst other features). The spatial features of the data do not change between records, while the time feature does as we are observing the change in features over time at a particular location.

The interpretation of the raw SCATS specific datasets is largely derived from definitions provided in the *SCATS 6.4.1 Operating Instructions* RTANSW (2004). The strategic monitor (SM) and volume store (VS) data are provided as binary files and converted into human readable comma separated value text by the proprietary *SCATS Traffic Reported Program* provided by the *Transport Systems Centre, Flinders University* (TSC).

The main data introduced here are:

- Traffic signal location data describing the location and layout of signalised intersections within SCATS as used in Adelaide.
- Accident data describing the time, location, severity, and other features of vehicle accidents within Adelaide.
- SCATS data describing the operation and control of the SCATS system, logging signal cycle lengths, vehicle flow counts and the configuration of the SCATS implementation.

The dataset and tools devised and described in this chapter are used throughout the research in following chapters, and made available within the TSC at Flinders University for future research in traffic engineering and elsewhere within the university for timeseries analysis and graph data research. Unfortunately due to the privacy demands of SCATS and DPTI, these data and tools cannot be made publicly available and are thus only provided by request subject to appropriate authorisation.

Although there is no personal information within the accident dataset, it remains private, as the time and location of incidents may potentially reveal personal details about the persons involved. DPTI do maintain an updated dataset without the exact time and date of each accident via the Data.SA initiative here: https://data.sa.gov.au/data/dataset/road-crashes-in-sa.

Due to the proprietary nature of the SCATS signalling and traffic flow datasets, formats, and tools, they are provided to the TSC commercial-in-confidence and thus cannot be shared publicly.

2.1 Data Storage

The majority of data used in the research in this thesis is stored in MongoDB, a free and open source document oriented database system developed by MongoDB Inc. Data in MongoDB are organised into databases which have many collections. Each collection has many documents. Each document is a BSON object (MongoDB 2018), an extended version of the JSON data format used to describe data in the JavaScript programming language. The subset of available BSON types we are concerned with are:

Object a mapping from a string key to a value. Values can be any valid BSON value. These use the format `{"key": value}`, where the object is enclosed in curly braces, the key is between double quotes and the key and value are separated by a colon.

Array a sequence of BSON values. These have the format `[value1, value2,... valueN]`, where the values are between square brackets and values are comma separated.

String a sequence of characters. They have the format `"characters"`, where the characters are between double or single quotes.

Integer an integer number. They have the format `1` or `-1`.

Float a floating point value representing a real number. These have the format `1.2345` or `-1.23456`.

Boolean either `true` or `false`.

null a `null` value.

Date an object that stores information about a date and time, optionally including information about its timezone. These have the format `ISODate("2016-01-01 00:00:00.000")` with values in the order: year, month, day, hour, minute, seconds and milliseconds.

MongoDB can also create indexes on its documents to allow fast querying over the documents within a collection; by default an index is created on an `_id` field which must be unique for every document in the collection. For example, if there is a collection of documents that contain a `site_no` field, and a query is performed to find those documents that have $site\_no$ = "3001", the database needs only to look in the index for the position on disk of those documents. This index uses a B-tree data structure, which takes on average $O(\log(n))$ comparison operations for search. For example, if a collection contains 1 billion records along with an index on some field, a search for a particular value in the index will require only 20 comparison operations.

When no index is present, the database must check each document in the collection for a match on `site_no`, which can take a significant amount of time [$O(n)$ on average as it is linear with respect to the number of elements in the collection] if there are millions or billions of individual documents. Thus, care must be taken as to create indexes that aid in the performance of queries that are expected to be performed on each collection so as to not consume excessive amounts of memory or computation time executing long running queries that must manually scan or sort an inordinate number of documents.

2.2 SCATS Data

In South Australia, the *Department of Planning, Transport and Infrastructure* (DPTI) is tasked with (among other duties) managing road traffic control signals. They use SCATS as their traffic control system and infrastructure.

The SCATS data provided by DPTI is collected by an inductive loop detector sensor beneath the road for each lane coming into an intersection where each sensor is uniquely identifiable on a per site basis. An image of one such detector as visible from the road is shown in Figure 2.2 and Figure 2.1, the black lines on the road outline the physical detector as it lies beneath the surface.

Klein et al. (2006) describe the principal by which these inductive loop detectors (ILDs) operate. The ILD is a loop of insulated wire with an alternating current passing through it, which induces a magnetic field around the wire. The presence of a vehicle or other sufficiently large metal object over this loop will distort this magnetic field which in turn reduce the inductance of the circuit. This change in the circuit is detected by a controller and when this change exceeds a specified threshold, the detector is marked as occupied until such time as the voltage of the circuit returns to normal, at which the point the detector is marked as unoccupied. Care must be taken in order to configure the threshold of the sensor appropriately for its application, as it may not be ideal for objects such as a pedestrian with coins or a phone to trigger the detector, but must still be able to register bicycles, motorbikes or scooters. At the other extreme, it may be required that the sensor only activates on large vehicles such as buses or trucks.

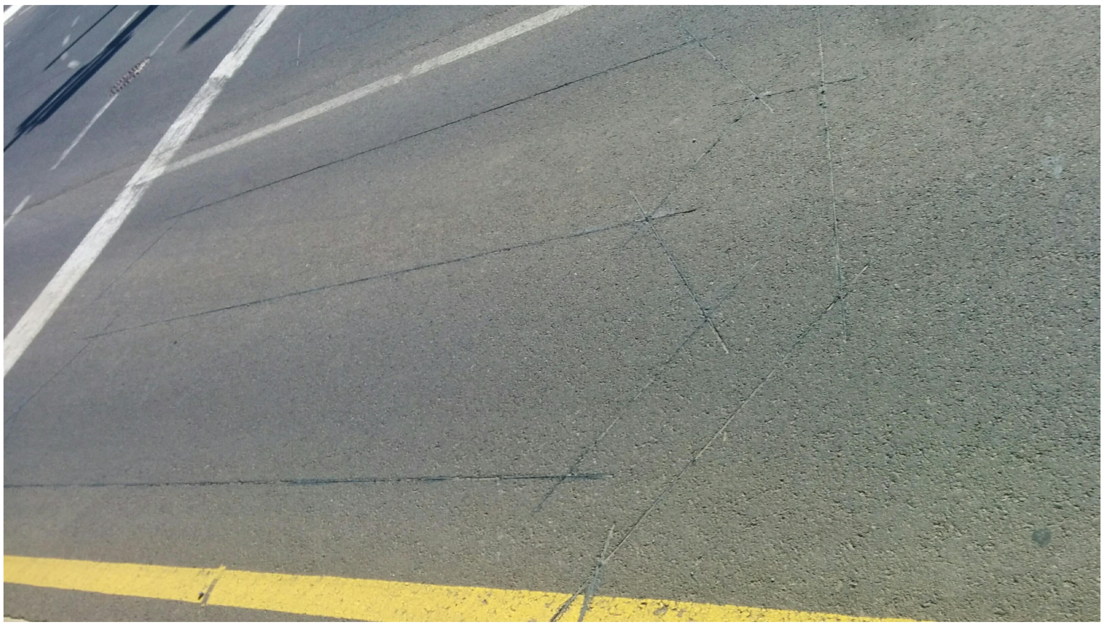


Figure 2.1: Close up photograph of loop detector showing the outline of the loop

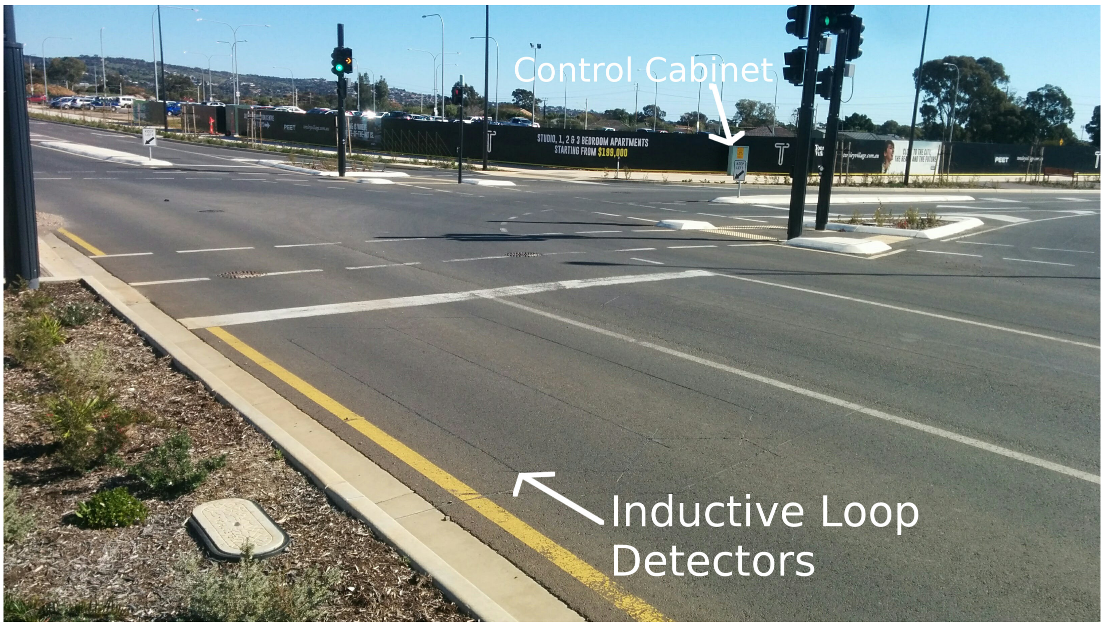


Figure 2.2: Photograph of a typical intersection with loop detectors (foreground) and control cabinet (background)

Such loop detector systems can have numerous issues which become present in the output data and effect data quality. These issues are (as seen in Figure 2.4):

- Missing data where the sensor does not report occupancy regardless of the presence of a vehicle
- Flickering where the sensor is either misconfigured or malfunctioning and reports more vehicles than actually pass over it. It is possible that a low rate

of flickering causes a valid traffic flow count (ie. less than 255), depending on the time and duration of this flickering, can be difficult to detect and may impact the quality of models that learn from it

- Constantly active where a malfunctioning sensor is always marked as occupied.

Traffic engineers may elect to ignore sensors in the last two cases so that their outputs are not used by the signal control system. Such error states must be corrected manually by sending a technician to the site which requires use of limited human time and equipment resources. As such, periods where erroneous values are being generated may last for weeks or months at a time. Because of this, care must be taken to clean the data and detect and ignore these values (where they can be reasonably identified) when implementing signal controllers and associated models.

In addition to managing traffic signals, SCATS also produces a number of files as a record of its configuration and activity on a daily basis. This research is concerned only with the following subset of its outputs:

- Region Configuration (LX) files: text files describing the organisation of sensors, intersections and cycle plans on a daily basis.
- Volume Store (VS) files: binary files containing vehicle count data aggregated into 5 minute blocks for every sensor at each intersection.
- Strategic Monitor (SM) files: binary files that contain reports about every phase and cycle at each intersection

Each region covers a number of intersections which are linked physically by roads, and within the SCATS system by links. There can be links between intersections in different regions, although these are in the minority and must be marked specifically as inter-region links when configuring the intersection (RTANSW 2004).

2.2.1 LX Data

Each file describes the intersections within a SCATS region for a day and how the signalling planner associates them, providing the following:

Intersection - information about an intersection and its internal communication configuration.

Cycle plans - each intersection has 4 plans and each is described in terms of the ordering and duration of each phase.

Phases - the discrete movements permitted at each intersection.

Strategic inputs - the sensors that are logically grouped together at each intersection. Each strategic input has 1 to 4 sensors.

Strategic approaches - a strategic input and the downstream phase associated with it.

The main goal of this work is to effectively analyse the SM data, specifically the *strategic input* field, the strategic inputs and their associated sensors at each intersection must be identified. This information is to be stored in the *locations* collection in the database. The LX data files have many data fields as described in *SCATS 6.4.1 Operating Instructions* (RTANSW 2004, Chapter 15), the ones of concern to this research work are:

- Intersections, which use the format:
  `SLOT10=6,1,4!INT=3001!VC=5!CS=273!PK=/ZSL=0!`, this tells us that slot 10 (denoted by `SLOT10`) of the SCATS system is used by intersection 3001 (denoted by `INT=3001`). The other fields are not used in this research.

- Strategic Approach, which use the format:
  `SA121=20!S^=121!VF=8,15**!VK=0!SD=3001A10C10F0!`, that is, strategic approach 121 (denoted by `SA121`) is part of intersection 3001 (denoted by `SD=3001`) and uses strategic input 121 (denoted by `S^121`). These are useful because they can be used to link strategic inputs with intersections. The remaining fields such as `VF` and `VK` are not used in this research.

- Strategic Inputs, which use the format:
  `SI121=3001,2!D#=6-7!`, that is, strategic input 121 (denoted by `SI121`) for the region are sensors 6 and 7 (denoted by `D#6-7`) of intersection 3001 (denoted by `SI121=3001`). Other strategic inputs may use the format: `D#1,3-5` to indicate that the SI uses sensors 1, 3, 4 and 5.

Algorithm 1 is presented to analyse the LX files in order to determine:

- Which intersections use which strategic inputs

- Which sensors each strategic input indicates

Additionally, a history of changes to strategic inputs at intersections is created, so that during data retrieval, VS data for a SI can be correctly generated by fetching the correct sensor values of an SI for any given day.

Algorithm 1: LX Intersection to Strategic Input Extraction Algorithm

```
input  : lxtext the content of the .lx file
output: Map of Intersections for region to their strategic inputs and
        associated sensors
1  intersections ← empty map;
2  intersection_names ← all matches for regular expression INT=(\d+)! in
     lxtext;
3  for intersection ←intersection_names do
4      intersections[intersection][‘strategicInputs’] ← empty map;
5  end
6  lines ← lxtext split into lines;
7  for line ← lines starting with ‘SI’ do
8      intersection ← intersection number from line;
9      si ← empty map;
10     si[‘site_no’] ← intersection;
11     si[‘sensors’] ← list of sensors following ‘#=’;
12     strategic_inputs[intersection] ← si;
13 end
14 for row ← lines starting with ‘SA’ do
15     strategic_input ← strategic input from line;
16     intersection ← intersection number from line;
17     intersections[intersection][strategic_input] ←
         strategic_inputs[strategic_input];
18 end
19 return intersections
```

For example, in the ACC region, the associations for intersection 3001 are:

```
"3001": {
     "121": {   "sensors": [6, 7],
                "site_no": "3001"},
     "122": {   "sensors": [2, 3],
                "site_no": "3001"},
     "123": {   "sensors": [13, 14, 15],
                "site_no": "3001"},
     "124": {   "sensors": [10, 11, 12],
                "site_no": "3001"},
     "125": {   "sensors": [4, 20],
                "site_no": "3001"},
     "6":   {   "sensors": [8],
                "site_no": "3001"}
}
```

Essentially, this means that intersection 3001 has 6 strategic inputs labelled 121, 122, 123, 124, 125 and 6. Strategic input 121 corresponds to the sensors labelled 6 and 7 of intersection 3001, strategic input 122 corresponds to sensors 2 and 3 of intersection 3001, and so on. Storing these associations then allows easy retrieval of grouped sensor counts from VS data, and to determine which sensors are being counted in the SM data. It is important to note that the intersection number is included (interchangeably referred to as 'site_no') as some intersections have strategic inputs from neighbouring intersections. For example, intersection 3056 has strategic input 188 that uses sensors from intersection 3020:

```
"3056" :{
    "188": {"sensors": [9, 10, 11], "site_no": "3020"},
    "29": {"sensors": [12, 13], "site_no": "3056"},
    "31": {"sensors": [5, 6], "site_no": "3056"},
    "33": {"sensors": [2, 3], "site_no": "3056"}}
```

Additionally, these strategic input configurations are subject to change by traffic engineers, such that the sensors used in a strategic input can be added or removed. The reasons for this are usually that a section of road has been physically modified, traffic demand has changed and so an engineer has modified the signalling, or an entire signalised intersection has been decommissioned.

To ensure that the correct sensor readings can be retrieved from the VS data for any particular day, LX files were parsed for every day and any changes to strategic inputs were added to a list for each intersection along with the day it changed. For example, here is the a subset of the history for intersection 3056:

```
[{
"date" : "20070101",
"si" : {
    "28" : {"site_no" : "3056","sensors" : [ 1, 2, 3]},
    "29" : {"site_no" : "3056","sensors" : [ 11, 12, 13]},
    "30" : {"site_no" : "3062","sensors" : [ 1, 2, 6, 7]},
    "31" : {"site_no" : "3056","sensors" : [ 4, 5, 6]},
    "32" : {"site_no" : "3066","sensors" : [ 1, 2, 3]},
    "33" : {"site_no" : "3056","sensors" : [ 8, 9, 10]}}
}, {
"date" : "20070217",
"si" : {
    "28" : {"site_no" : "3056","sensors" : [ 1, 2, 3]},
    "29" : {"site_no" : "3056","sensors" : [ 11, 12, 13]},
    "30" : {"site_no" : "3062","sensors" : [ 1, 2, 6, 7]},
    "31" : {"site_no" : "3056","sensors" : [ 5, 6]},
    "32" : {"site_no" : "3066","sensors" : [ 1, 2, 3]},
    "33" : {"site_no" : "3056","sensors" : [ 8, 9, 10]}}
}, {
"date" : "20100708",
"si" : {
    "28" : {"site_no" : "3056","sensors" : [ 2, 3]},
    "29" : {"site_no" : "3056","sensors" : [ 11, 12, 13]},
    "31" : {"site_no" : "3056","sensors" : [ 5, 6]},
    "33" : {"site_no" : "3056","sensors" : [ 9, 10]}
}]
```

That is, sensor 4 was removed from strategic input 31 on 2007/02/17. On 2010/07/08 strategic inputs 30 and 32 were removed along with with sensor 1 from SI 28 and sensor 8 from SI 33.

2.2.2 Strategic Monitor Data

Strategic Monitor (SM) data stores information about every executed phase at an intersection during operation and exported to a file at the end of each day. The exported files are parsed using the *SCATS Traffic Reporter* program, the outputs as defined in *SCATS 6.4.1 Operating Instructions* for every cycle of which we are concerned are:

Timestamp a record of when the phase occurred (without seconds), thus we must sort by sequence (the order in which the records appear in the SM file) in order to get the actual sequence of phases.

Site number the intersection identifier

Strategic approach for each strategic approach at the intersection:

- **Strategic approach number** the identifier of the strategic approach
- **Phase mask** the movements permitted during the phase
- **Phase time** duration in seconds of the phase
- **Measured flow (VF)** the total number of vehicles counted over the sensors of the strategic approach during the phase
- **Reconstituted flow (VK)** a smoothed number of vehicles on the strategic approach during the phase in order to prevent frequent switching between cycle plans for high and low volume observations. Calculated as the average of the previous volume (VF_{t-1}) over specified sensors and the vehicles per hour (VPH) divided by 20, or $\text{VK}_t = \left(\text{VF}_{t-1} + \frac{\text{VPH}}{20}\right)/2$.

Data was imported into MongoDB from all SCATS regions in the period 1 January, 2012 to 13 December, 2017. Each record contains the data for 1 phase, for example:

```
{
    "datetime" : ISODate("2016-01-01 00:00:00.000"),
    "phase_time" : 30,
    "site_no" : "376",
    "strategic_input" : 55,
    "measured_flow" : 0,
}
```

That is, at midnight on 1 January 2016, the strategic input 55 at intersection 376 measured 0 flow and the phase lasted for 30 seconds. There are 11,214,646,724 (between 1/1/2012 and 4/6/2018) such records stored in the database with a compound index on `site_no`, `strategic_input` and `datetime`, allowing performant queries over a strategic input at a particular site in a given time period ordered by their occurrence.

2.2.3 Volume Store Data

Volume store (VS) data stores data for each sensor at each intersection for the following 5 minutes and are parsed using the *ndp* program provided by the TSC.

Datetime the date and time when the period starts

Site Number the intersection at which the reading was taken. Site numbers are an abstraction and do not indicate that two sites are in connected or indicate the distance between sites.

Sensor the unique identifier for a particular sensor at a site which typically corresponds to one lane of traffic. The majority of sensors are loop detectors, but may be from other sources, such as cameras. Each site can have up to 24 sensors.

Vehicle count the number of vehicles detected over the sensor during the period

The SCATS data was inserted into a MongoDB collection with each document having the following layout:

```
{
    "site_no" : "3060",
    "datetime" : ISODate("2006-10-13T23:55:00.000+0000"),
    "readings" : {
        "1" : 2047,
        "2" :3,
        "3" :2047,
        ...
        "24" :0
    }
}
```

That is, on 13 October, 2006, at 11:55 PM, at intersection 3060, sensors 1 and 3 were in error state, and sensor 2 recorded 3 vehicles. All other sensors recorded 0 vehicles or were not connected. Since each site can have up to 24 sensors, missing sensors are 0.

Each document in the collection covers five minutes of traffic flow at a particular site. This data is stored with indices on `datetime`, `site_no` and a compound index over `datetime` and `site_no`. This allows for fast querying for site over any given time period.

The sensor counts may also be in an error state (RTANSW 2004):

2046 if a sensor has a detector alarm (DA). DA is caused by the detector not changing states between occupied and unoccupied.

2047 if the intersection does not return a VS volume after 4 requests by SCATS.

While not described by RTANSW (2004), the sensor may also be in a flickering state where it reports a vehicle count much higher than the number of vehicles that physically passed over it. This can show as a reading of 256, the largest possible flow that VS can store for a sensor (ie. VS stores flows as an unsigned 8 bit integer). Given that there are 300 seconds in 5 minutes, a VS readings of 256 would mean 1 vehicle passing over the sensor every 1.17 seconds continuously for the 5 minute duration. This event is extremely unlikely and thus must be filtered out before analysis can be performed. Additionally, the sensor may be flickering and report a number of vehicles below 256, but is still far beyond the average of other sensors in the strategic input. These types of erroneous values are shown in Figure 2.3.

2.2.3.1 Generating a Dataset

There are two forms of data that can be extracted from the VS data:

Sensor flow - this is the flow data over a group of one or more sensors at intersection. Its retrieval is trivial as the VS dataset can be queried by the intersection and date range required. The resulting data is the flow values of desired sensors in the returned documents.

Strategic input flow - this is the flow through a strategic input. Because strategic input configuration may change over time, VS data must be retrieved for each configuration period. Thus, Algorithm 2 has been devised to extract a subset of data for the experiments used in the following chapters.

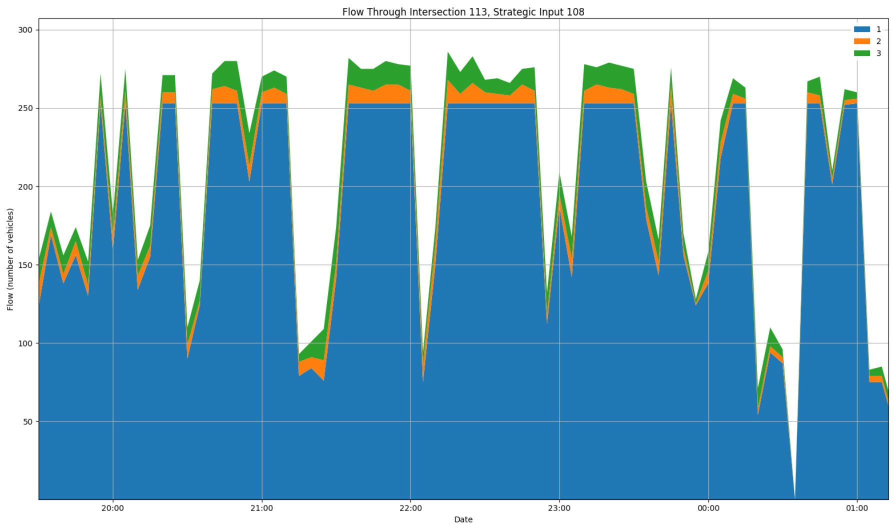


Figure 2.3: Stackplot of sensor readings 1,2 and 3 at intersection 113. Sensor 1 is flickering during this period, recording extremely high counts (especially considering the flows recorded between 7:30 PM and 1:30 AM) while sensors 2 and 3 report normal flows

Both of these datasets are filtered during generation to exclude error values which can drastically affect the distribution of the data. For example, intersection 113, strategic input 108 has a change in distribution as seen in Figure 2.4. This particular strategic input does not change configuration over the period and only records sensors 1, 2 and 3. Plotting the flow of each sensor individually shows that at many times, some sensors fall into an error state for extended periods. Additionally, sensor 1 frequently goes into the 'flickering' state.

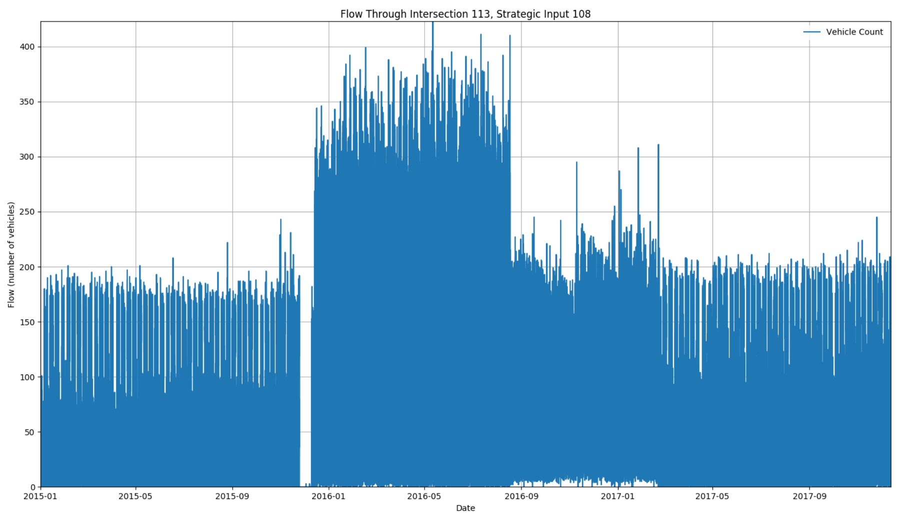


Figure 2.4: Total flow through intersection 113, strategic input 108

Algorithm 2: Method to Generate a dataset containing the flow for a strategic input at an intersection

```
input  : intersection, strategic_input
output: List of timestamps and flow for that timestamp for the given
        strategic input
1  siConfigurations ← strategic input configurations for
     intersection=$intersection$;
2  sort $siConfigurations$ by date;
3  flows ← empty list;
4  for $n = 1$ to $length(siConfigurations) - 1$ do
5      conf1 ← $siConfigurations_n$;
6      conf2 ← $siConfigurations_{n+1}$;
7      si ← $conf1_{strategic\_input}$;
8      for $s \leftarrow 1$ to $length(si)$ do
9          $sensor \leftarrow si_s$;
10         readings ← get VS data sorted by timestamp where:
             $(conf1_{date} \leq timestamp < conf2_{date}) \wedge (site\_no = sensor_{site\_no})$;
11         for $r \leftarrow 1$ to $length(readings)$ do
12             flow ← 0 ;
13             reading ← $readings_r$;
14             for $s \leftarrow 1$ to $length(si_{sensors})$ do
15                 sensorFlow ← $readings.sensors_s$;
16                 if $sensorFlow$ is not an error value then
17                     flow += sensorFlow;
18                 end
19             end
20             flows.append([reading.datetime, flow]);
21         end
22     end
23 end
24 return flows
```

2.3 Traffic Signal Location Data

This is derived from Geographical Information Systems (GIS) shape files provided by the Flinders TSC about each set of traffic signals:

- The geographical coordinates, local government area (LGA) and SCATS region
- Description of the signals (ie. which road(s) the signals lie on)
- The type of signals (signalised intersection or pedestrian crossing)

- The intersection code (eg. TS3001)

The data has been extended with data described in subsection 2.2.1, each intersection has been inserted into the database in the following format, including additional fields as derived in this chapter:

```
{
"site_no" : "157"
"loc": {"type": "Point", "coordinates": [ 138.59784, -34.81626]},
"type": "Traffic Signals",
"description": "* 3-ARM * Port Wakefield Rd RN3500-Salisbury Hwy RN5406",
"scats_region": "WALKVL",
"lga": "CITY OF SALISBURY",
"scats_diagram": "iVBORw0KGgoAAA...",
"neighbours": [ "156", "8", "344"],
"strategic_inputs": [ {
    "date": "20080310",
    "si": {
        "264": {
            "site_no" : "157",
            "sensors" : [ 1, 2]},
        "265": {
            "site_no" : "157",
            "sensors" : [ 3, 4]}}
    }
]}
```

`scats_diagram` is a base 64 encoded image of the SCATS diagram for the intersection, an example is shown in Figure 2.5

`lga` is the Local Government Area in which the intersection is located

`loc` is a GeoJSON Point indicating the location of the intersection in longitude and latitude

`strategic_inputs` is an array of historical strategic input configurations for this intersection

`neighbours` is an array of `site_no` for intersection that are immediately upstream or downstream of this intersection

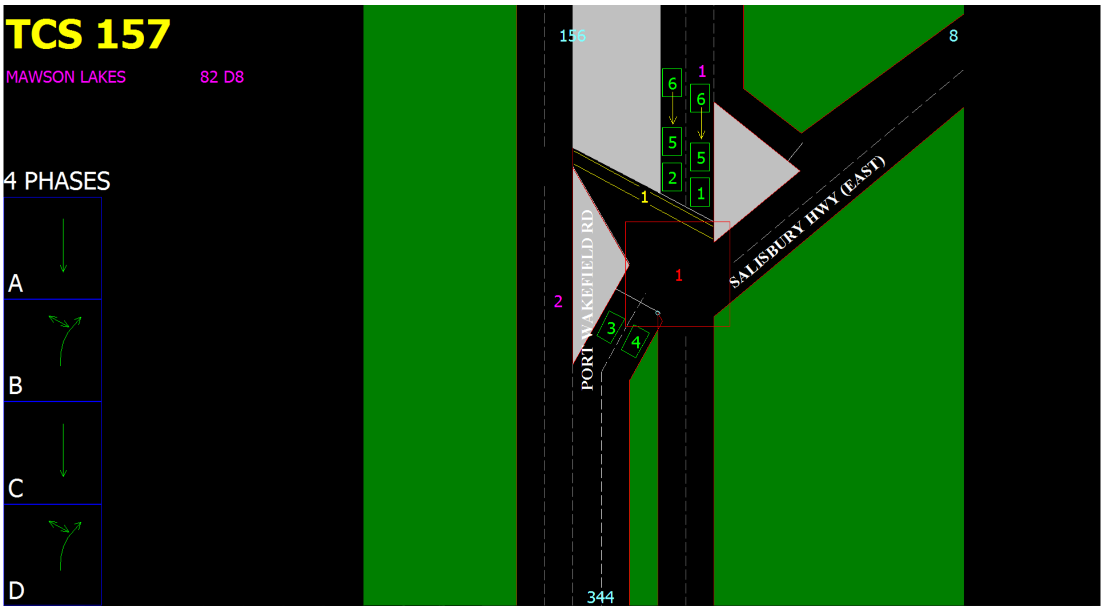


Figure 2.5: SCATS diagram for intersection 157

2.4 SCATS Diagrams and Turning Movements

Previous work (Zhang 2005) has shown improved incident detection performance when upstream and downstream flows are provided as model inputs in addition to link flow. To construct such inputs for the models used in this work, a dataset was created that stores for each intersection:

- The upstream sensors for a strategic input
- The downstream intersection for a sensor

Having such data enables the modelling of the Adelaide road network as a graph where the nodes are intersections and edges are the upstream and downstream sensors of a link. This shown in Figure 2.6, where intersection 3001 is shown with all other intersections that are at most 2 link journeys away and arrows indicating the direction of permitted travel between intersections.

The data is not readily available from the SCATS system, and thus an appropriate dataset was generated by extracting the sensor and neighbour intersection data from SCATS diagrams generated by the 'SCATS Picture' program. An additional program was written that could quickly perform the repetitive task of using this application to retrieve the SCATS diagram for every intersection. Both applications are shown in Figure 2.7.

SCATS diagrams (examples shown in Figure 2.8) provide the following information:

- Intersection name at top left, eg. TCS 3084

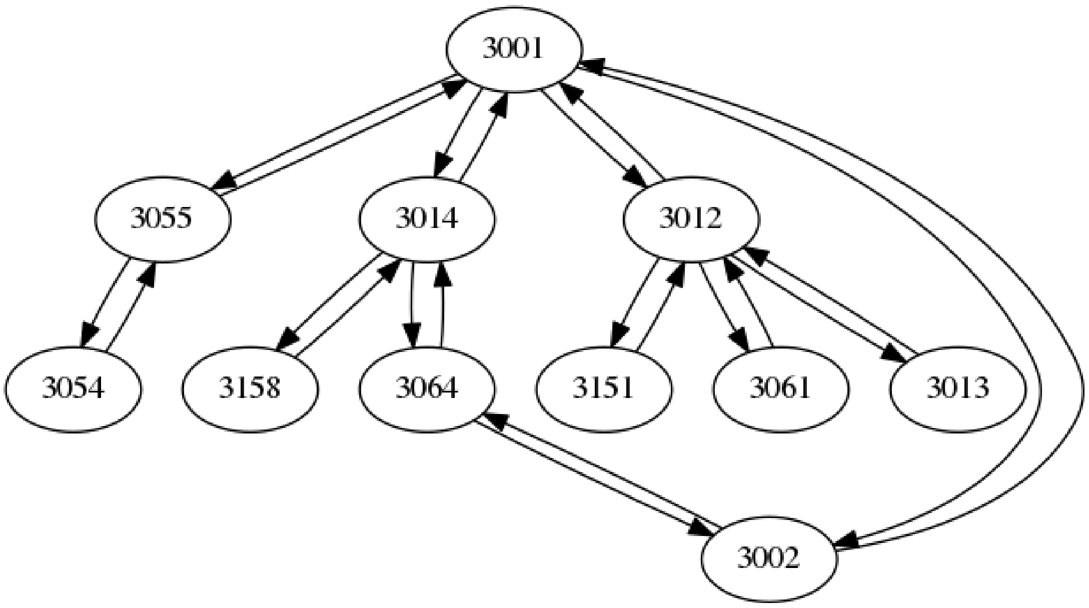
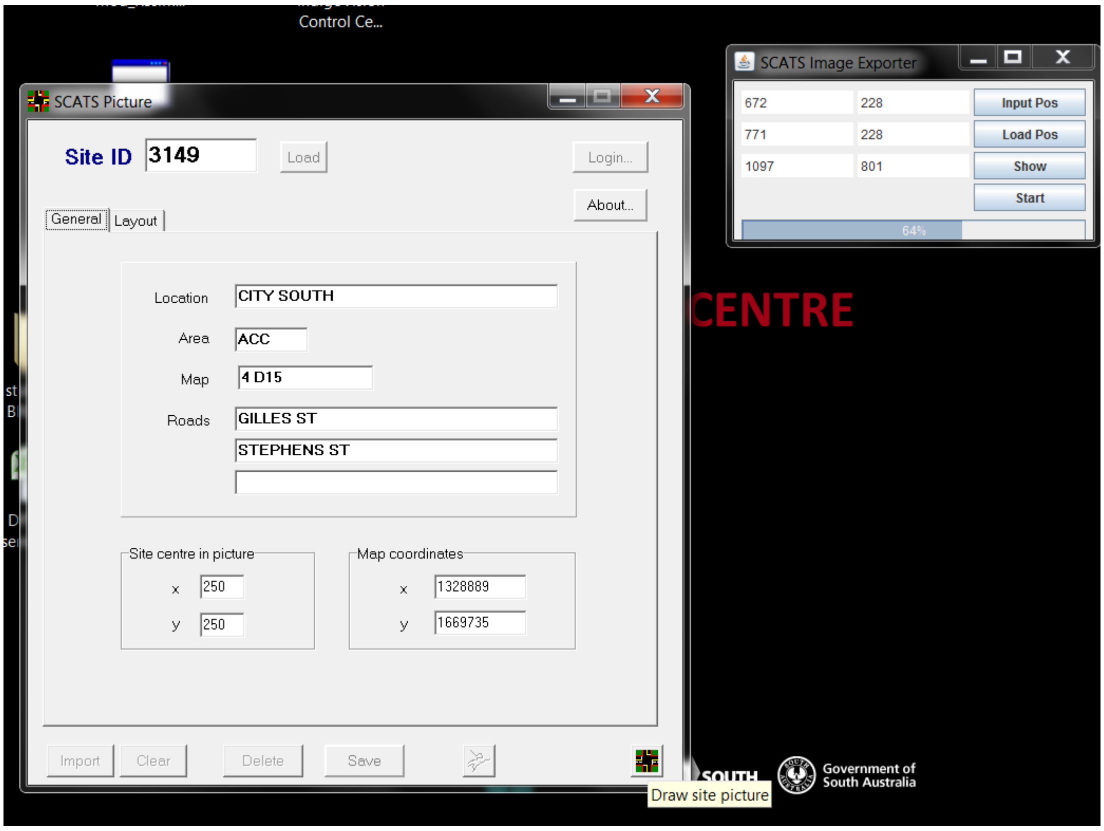


Figure 2.6: Subset of the network of intersections, with intersections that are most 2 steps from 3001



SCATS Picture

Site ID 3149

Load Login... About...

General Layout

Location	CITY SOUTH
Area	ACC
Map	4 D15
Roads	GILLES ST
	STEPHENS ST

Site centre in picture

x	250
y	250

Map coordinates

x	1328889
y	1669735

Import Clear Delete Save

SCATS Image Exporter

672	228	Input Pos
771	228	Load Pos
1097	801	Show
		Start

64%

Figure 2.7: 'SCATS Picture' and 'SCATS Image Exporter' programs

- SCATS region
- SS indicates the SCATS subsystem identifier. SCATS subsystems are log-

ically grouped collections of intersections for the purpose of coordination (RTANSW 2004)

- Discrete intersection phases and movements available during those phases.
- The illustration of the intersection intended to be an approximation of its physical layout.
- The numbered parallel yellow lines indicate pedestrian crossings
- Green areas indicate areas that are not road
- Black areas indicate road
- Grey areas indicate traffic islands (areas excluded from traffic, as described by ARRB (2015))
- Green boxes with a number inside indicate a detector and its identifier. When the detector is occupied, the box is filled with blue and the number turns white
- White dashed lines indicate the lanes of the road
- White words indicate the name of the road
- Blue numbers near the image border indicate the name of the neighbouring intersection
- Magenta numbers indicate the traffic signal group number. Signal groups are collections of individual lanterns on the signals for a specific movement.

For example, in Figure 2.8, the sensors labelled 6,7,8,9,13 and 14 at intersection 3084 are upstream of intersection 3043's sensors on its south side labelled 5,6,7,8,9 and 10.

In order to extract this information from the files provided, then for each neighbouring intersection with a provided SCATS diagram the upstream sensor numbers were extracted using the method described in algorithm 4. Multiple methods were used to extract information, with optical character recognition over contour enclosed regions of a heavily preprocessed image working better than template matching.

2.4.1 Extracting Sensors with Template Matching

Template matching (Lewis 1995) is the process whereby a small template image is made from a reference image which is then passed over a query image and an array of matching scores at each location in the image returned. Each template is then moved over a simplified grayscale version of the image. Where template

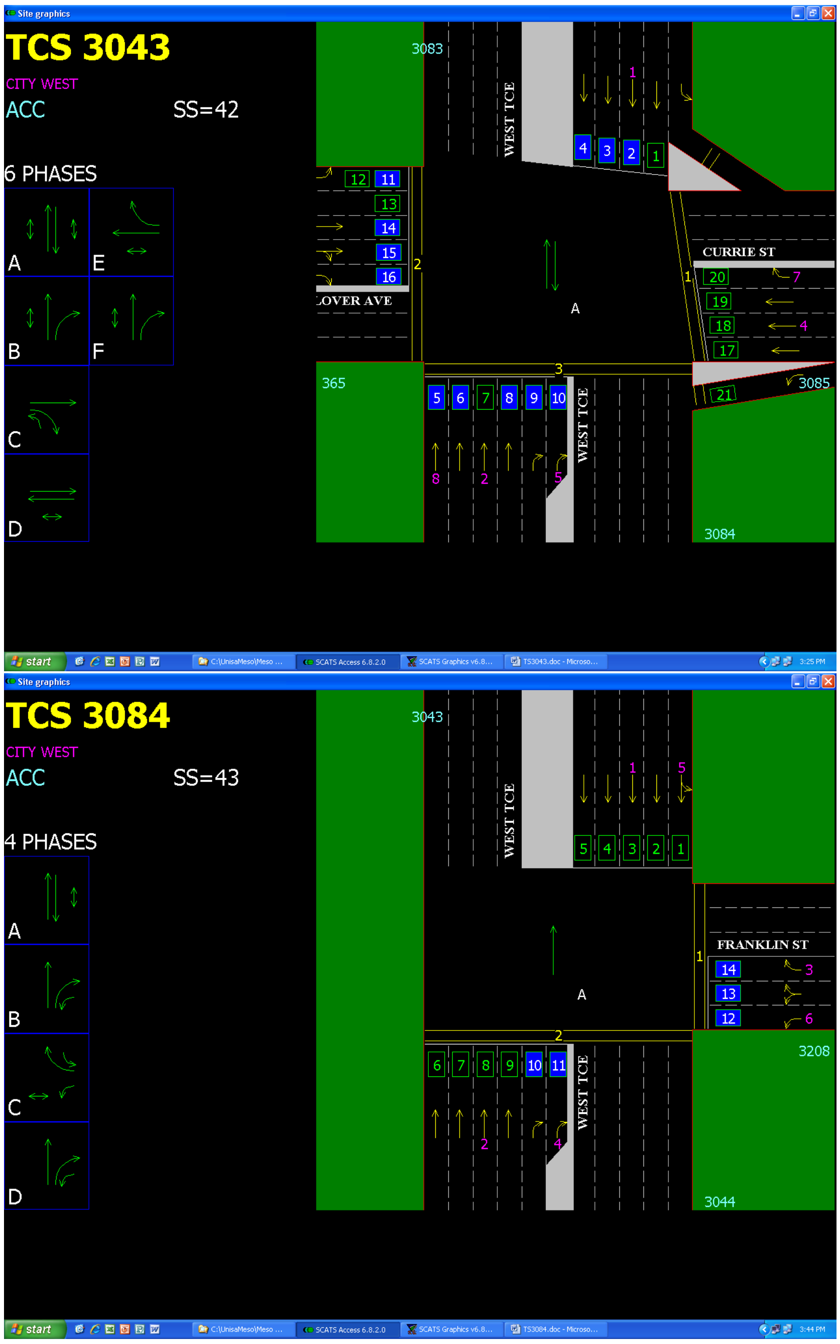


Figure 2.8: Intersections 3043 and 3084

matches overlap, the match with the highest match score and longest label name are used (this removes matches where a single digit is matched over a double digit number eg. 1 and 16 or 8 and 18).

To determine the downstream sensors for a particular link, extracting the arrows from the SCATS image was attempted using template matching, which involves the following steps:

1. Crop an image to a given area to select the template

2. Slide the template over the image you wish to find the template in and calculate a score for how well the template matches over the current target area.
3. Return the template matching scores
4. The match scores can then be filtered by a threshold to determine if there are several matches

In practice this method was unreliable and slow, namely because the arrows used in the SCATS diagrams are inconsistently drawn and so would often match.

2.4.2 Extracting Sensors with Contour Detection and Optical Character Recognition

Optical character recognition is the process of extracting a text string from an image. Many methods exist to do this Google's Tesseract OCR library: libtesseract, (Smith 2007) was selected because of its ease of use and free and open source status.

Contour detection in image processing is the process of extracting contours. Suzuki & Be (1985) describe the border following method as implemented in OpenCV OpenCV (2015) which was used for the implementation.

The following steps were used to extract sensor locations and labels from the intersection images:

1. Convert the image to grayscale to remove colour
2. Find contours in the image that are approximately the same size as those sensor labels in the image
3. Perform OCR on the region enclosed by the contour
4. Return the results of OCR along with their location in the image

2.4.3 Algorithms Used

Although template matching was successful on the image the templates were extracted from, these templates did not accurately match to arrows in other SCATS diagrams. Because of these limitations, data was automatically extracted from a spreadsheet provided by TSC to determine upstream sensors in the Adelaide City Council (ACC) region.

To determine downstream sensors, OCR was used to extract sensor labels from contours that enclosed regions of the image with a specific size as described in Algorithm 3.

These clusters of sensors are then joined with the neighbouring intersections as described in Algorithm 4

Algorithm 3: Function for extracting groups of sensors at a given intersection

```
   Input: image of intersection, intersection
   Output: Clusters of sensors for intersection
1  Assume image is represented using Blue,Green,Red format with each
     channel's intensity in the range 0,255;
2  Convert $(0, 127, 0)$ and $(0, 0, 255)$ coloured pixels in image to black;
3  Convert image to grayscale;
4  $SensorPositions \leftarrow$ empty list;
5  /* Extract the position of each label from the image        */
6  $contours \leftarrow$ findContours(image);
7  foreach $contour \leftarrow contours$ do
8      if $26 \leq contour_{height} \leq 29$ and $39 \leq contour_{width} \leq 42$ then
9          subImg $\leftarrow$ section of image enclosed by contour;
10         remove border pixels from subImg;
11         apply a binary threshold to subImg;
12         sensorLabel $\leftarrow$ OCR(subImg);
13         /* Each sensor has its label and x,y positions      */
14         SensorPositions.append([sensorLabel, $subImg_x, subImg_y$]);
15     end
16 end
17 /* Cluster the sensors                                       */
18 $clusters \leftarrow DBSCAN(SensorPositions, minPts = 1, \epsilon = 80)$;
19 foreach $cluster \leftarrow clusters$ do
20     $min_x, min_y \leftarrow$ minimum x,y values from $cluster$;
21     $max_x, max_y \leftarrow$ maximum x,y values from $cluster$;
22     $cluster_{isHorizontal} \leftarrow abs(min_x - max_x) > abs(min_y - max_y)$ ;
23     $cluster_{xMedian} \leftarrow median($cluster x values$)$ ;
24     $cluster_{yMedian} \leftarrow median($cluster y values$)$ ;
25 end
26 return $clusters$
```

Algorithm 4: Function for associating sensor clusters with neighbouring intersections

```
Input: clusters of sensors for intersection, intersection
Output: For each neighbour of the intersection, the set of sensors that
        monitor traffic flow from the neighbouring intersection to the
        intersection being analysed
1  /* neighbours is an associate array mapping north, east,
      south, west to their respective neighbouring intersection,
      null if no neighbouring intersection exists in that
      direction                                                    */
2  $neighbours \leftarrow getNeigbouringIntersections(intersection)$;
3  /* Associate each cluster of sensors with a neighbouring
      intersection                                                 */
4  foreach $cluster \leftarrow in\ clusters$ do
5      if $cluster_{isHorizontal}$ then
6          if $image_{height} - cluster_{yMedian} < image_{height}/2$ then
7              $intersection_{neighbours['south']} \leftarrow cluster$
8          else
9              $intersection_{neighbours['north']} \leftarrow cluster$
10     else
11         /* Differentiate by distance from middle of image          */
12         if $image_{width}/2 - cluster_{xMedian} < image_{width}/4$ then
13             $intersection_{neighbours['east']} \leftarrow cluster$;
14         else
15             $intersection_{neighbours['west']} \leftarrow cluster$;
16 end
17 return $intersection$
```

2.5 DPTI Accident Data

DPTI has provided a dataset of all 142,514 reported vehicular accidents in the Adelaide metropolitan area for the period 1/01/2006 to 1/2/2015. Each record contains:

- Date and time
- Street address
- Road speed limit at point of accident
- Cause of accident
- Road type
- Whether or not a four-wheel drive vehicle was involved

- Current weather conditions
- Road moisture level (wet or dry)
- Crash type
- Other remarkable features of the crash site
- Exact location (latitude and longitude)
- Approximate dollar value of damages
- If traffic controls were being used
- Number of vehicles involved
- Severity, one of:
  - Fatality (F)
  - Serious Injury (SI)
  - Minor Injury (MI)
  - Property Damage Only (PDO)

The possible weather conditions, types and causes of accidents are listed in Appendix B. These crashes were inserted into a MongoDB collection with an index on `datetime` and `location`.

The accidents recorded are only those reported to the police, as such the full dataset may not accurately reflect every single accident, as very low severity incidents may not be reported. Such potential gaps in the data are likely to be irrelevant to the problem of incident detection, as they are likely to be quickly cleared and not effect the traffic flow.

The vast majority of accidents are PDO (70.92%), while MI account for 25.91%, SI are 2.90% and fatalities are 0.26%. Future work on this data could look at correlations between traffic delay, crash type, severity and location in order to advise the targeted improvement of road safety initiatives.

2.6 Conclusion

The traffic flow, reported vehicle incidents, road network configuration datasets and associated algorithms presented in this chapter are used throughout the following chapters in this thesis and form the foundation of the models developed. Furthermore, the datasets introduced here are made available within the TSC at Flinders University could be readily used to facilitate teaching and future research work in the fields of traffic engineering and data science. It is unfortunate that this dataset is not available to the wider research community, but due to licensing restrictions placed on these datasets by DPTI and SCATS, this is not possible.

Approved researchers may use the database to easily extract traffic flows, network configurations, and accident data to:

- Understand and investigate sections of the road network and their usage over time
- Design new or updated road network configurations
- Investigate the nature of road incidents and their impact on traffic flows
- Develop more accurate traffic models based off the large volume of historic data for use with existing traffic simulation software
- Develop new algorithms for short term timeseries prediction models and evaluating them against the baseline results presented in the following chapters of this work.

Chapter 3

Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a collection of algorithms for modelling the processes of the mammalian neocortex. Experiments show that it can perform competitively compared to other state of the art time series prediction algorithms such as ARIMA and LSTM.

In this chapter, the HTM algorithms are described with their biological motivations as introduced by Hawkins et al. (2016) and utilised in Chapter 4 and Chapter 5.

3.1 HTM Model

HTM uses a model of the mammalian neocortex, a section of the brain made up of vertically aligned columns, with horizontal layers between them responsible for taking in sensory input (visual, auditory, tactile, etc.) and outputting signals for tasks such as motor function and object recognition (Byrne 2015, Mountcastle 1997). Subsequent layers are responsible for different outputs (eg. motor function, language) where regions (groups of columns) within each layer are sensitive to different inputs (Hawkins & Ahmad 2016). HTM seeks to implement a computational model of this theory of the neocortex.

HTM at a high level has 4 main steps (see Figure 3.1) (Hawkins et al. 2016):

1. Encoders are fed time series inputs such as scalars, categories, date/time, or location and output a Sparse Distributed Representation (SDR). A single field must be marked as a Predicted Field (PF).

2. This SDR is then fed to a Spatial Pooler (SP), which produces an SDR from its active columns. The SP attempts to learn the spatial properties (ie. the distribution of on-bits for similar encodings) of its input SDRs.

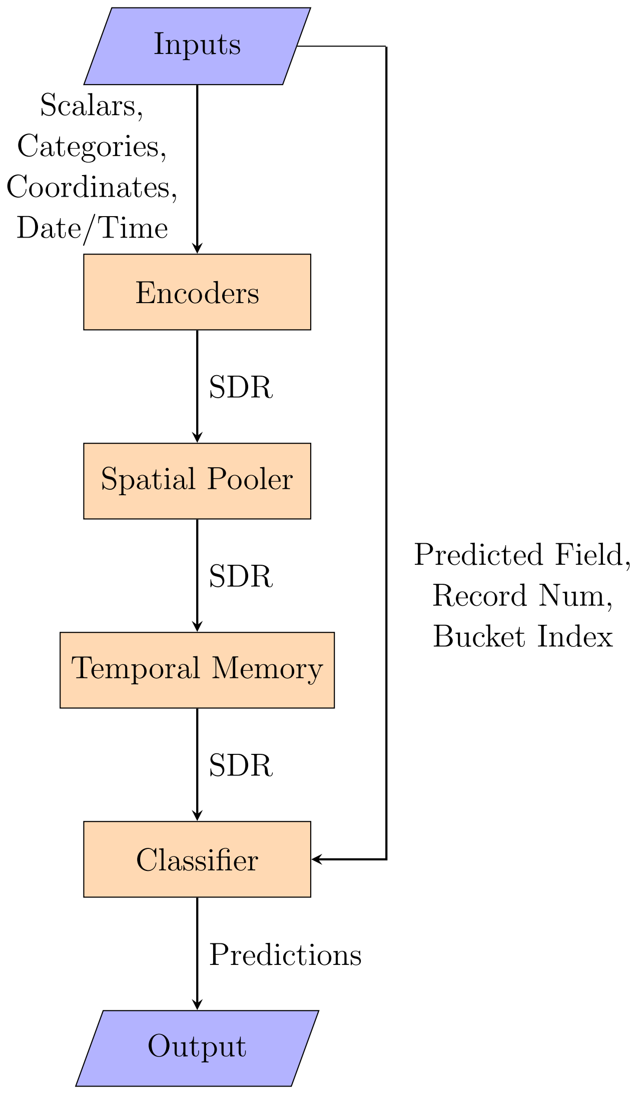

Inputs
Scalars, Categories, Coordinates, Date/Time
Encoders
SDR
Spatial Pooler
SDR
Temporal Memory
SDR
Classifier
Predicted Field, Record Num, Bucket Index
Predictions
Output

Figure 3.1: High level flow chart of HTM algorithm

3. This SDR is then fed to a Temporal Memory (TM) and produces an output SDR from its active columns. The TM attempts to learn the input sequence of SDRs over time.

4. This SDR is then fed to the Classifier along with the value of the PF, bucket index of the PF and the record number to give predictions from which anomalies can be inferred. The classifier will produce a probability distribution over the PF k steps into the future.

3.1.1 Sparse Distributed Representations

Empirical evidence suggests that the neocortex represents information using sparse distributed representations (Ahmad & Hawkins 2016). This means that for a given input to the neocortex, only a small subset of neurons will be activated.

Cognition would be impossible if a majority of neurons activated for any given input, for example, one subset of columns may have learned to activate when a 'light' visual input is detected in a certain part of vision. If other columns also activated that normally reacted to 'dark' or 'blurry' , one would be deeply confused as to what they were seeing. It is this behaviour that is emulated by SDRs, exploiting the fact that small regions of neurons are sensitive only to specific inputs. This indicates that the neocortex is a robust learner, using an ensemble of a large number of small models, each specialising in activating in response to a particular type of input. It is this property that SDRs and HTM attempts to emulate and take advantage of.

Within HTM, data is encoded using sparse distributed representations (SDR), Ahmad & Hawkins (2015) describe these as binary vectors of length n with a constant number w of ON bits[1]. Each input to the algorithm, typically a timestamp or location along with a scalar or categorical value, is encoded into an SDR. The w value is typically very small in comparison to n, so as to maintain sparsity (s) typically between 0.05% and 2%, defined by:

$$s = \frac{w}{n}$$

This low sparsity means that SDRs can cover a large input space, so given n and w the number of unique encodings is the binomial coefficient of n and w:

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

This small sparsity and large number of unique possible encodings also ensures that the probability of any 2 inputs, x and y, having the same representation is exceedingly small:

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

The overlap between any two SDR encodings can be calculated as their overlap score, that is, the number of common on bits (which is conveniently their algebraic dot product since the product of two overlapping zeros, or a one and zero will always be zero):

$$overlap(\mathbf{a}, \mathbf{b}) = \sum_{1}^{n} a_i b_i = \mathbf{a} \cdot \mathbf{b}$$

The simplest encoding scheme is simply to mark w continuous bits as 1 and the remainder 0. The ON bits are centred at index:

[1]Bits may be either ON (1), or OFF (0). The biological reality of neuronal activation levels is beyond the scope of this thesis.

$$centerIndex = \frac{input - minValue + \frac{resolution}{2}}{resolution} + padding$$

Where:

$$\begin{aligned} resolution &= \left\lfloor \frac{maxValue - minValue}{n - w} \right\rfloor \\ padding &= \frac{w-1}{2} \end{aligned}$$

The encoded output will have ON bits in the interval $[centerIndex - padding, centerIndex + padding]$.

As an example, suppose a field with possible values in the range $[0, 10]$, when encoding the values $[0, 10]$ into an SDR with $w = 5$ and $n = 32$ (calling such an encoding function E), the output is:

$$\begin{aligned} E(0) &= 11111000000000000000000000000000 \\ E(1) &= 00011111000000000000000000000000 \\ E(2) &= 00000111110000000000000000000000 \\ E(3) &= 00000000111110000000000000000000 \\ E(4) &= 00000000000111110000000000000000 \\ E(5) &= 00000000000000111110000000000000 \\ E(6) &= 00000000000000001111100000000000 \\ E(7) &= 00000000000000000001111100000000 \\ E(8) &= 00000000000000000000001111100000 \\ E(9) &= 00000000000000000000000011111000 \\ E(10) &= 00000000000000000000000000011111 \end{aligned}$$

It can be observed here, that similar values, eg. 0 and 1, are quite similar in their encoding and dissimilar values have very different encodings. In the above examples, $overlap(E(0), E(1)) = 2$ while $overlap(E(0), E(9)) = 0$. This is ideal for non-periodic data, for example:

- Counts such as number of vehicles,
- Readings such as temperature, pressure, or load.

When encoding cyclical data, such as:

- Day of week

- Hour of day
- Day of year

It is assumed that data with significant numerical distance, may have similar semantic distance. For example, if a numerical mapping is made from weekdays to numbers: $0 \rightarrow \text{Sunday}, 1 \rightarrow \text{Monday}, 2 \rightarrow \text{Tuesday} \ldots 6 \rightarrow \text{Saturday}$, while Sunday and Saturday have a numeric distance of 6, their actual distance is 1 when considering Sunday comes immediately after Sunday. To this end, an encoding is defined such that inputs near the minimum and maximum values have a higher degree of overlap than they would otherwise. In order to do this, the *centerIndex* becomes:

$$centerIndex = \frac{(input - minValue) \times n}{maxValue - minValue}$$

ON bit indexes outside the range $minValue \leq idx \leq maxValue$ are reassigned to $idx \mod n$ (where $\mod$ is the remainder from division). Example outputs for days of the week with $n = 32$, $w = 5$, $minValue = 0$ and $maxValue = 7$ to define the encoder E_p. The outputs for the days of the week are (where 6.5 is close to midday on the last day):

$$\begin{aligned}
E_p(0) &= 11100000000000000000000000000011 \\
E_p(1) &= 00111110000000000000000000000000 \\
E_p(2) &= 00000001111100000000000000000000 \\
E_p(3) &= 00000000000111110000000000000000 \\
E_p(4) &= 00000000000000001111100000000000 \\
E_p(5) &= 00000000000000000000111110000000 \\
E_p(6) &= 00000000000000000000000001111100 \\
E_p(6.5) &= 00000000000000000000000000011111
\end{aligned}$$

It can be observed here that $overlap(E_p(0), E_p(6.99)) = 4$, which is a significantly higher overlap than if the encoder were non-periodic.

In order to feed multiple inputs to a HTM, individual fields are encoded with separate encoding schemes and concatenated to create the final input. Care must be taken when specifying the number of ON bits in each encoding, given that fields with significantly greater w values will dominate the activations for that input. Although it may be discovered automatically through hyperparameter optimisation (or based on the model designer's experimentation, intuition or knowledge) that such domination/relegation would result in better model performance due to the field's significance in relation to the value being predicted.

For example, suppose there are two inputs, daily count and day of week. A periodic encoder would be used for day of week, and non-periodic encoder for count, using the above encoders E and E_p examples, the following encodings for inputs $day = 1 = Monday$ and $count = 4.5$ are:

$$\begin{aligned} E(4.5) + E_p(1) = &00000000000011111000000000000000 \\ &00111110000000000000000000000000 \end{aligned}$$

In practice, encoders are used with far larger n, and more sophisticated encoding schemes for more varied and composite datatypes. For example:

1. Geospatial data containing latitude, longitude, speed and altitude could be made into a single unique encoding by concatenating the scalar encodings of each respective field.
2. Categories can be encoded using a scalar encoder where each category is converted to an integer in the range 1 to the maximum number of encountered categories.
3. Logarithmically scaled data can be encoded such that values are similar as the order of magnitude of the value grows.

3.1.2 Spatial Pooler

The Spatial Pooler (SP) as described by Hawkins et al. (2016), takes as input an SDR and produces an SDR as output from its active columns, which aims to represent the minicolumns in the brain (Byrne 2015). The main goals of the SP are to:

1. Maintain a fixed sparsity in the output, this means that it will produce SDRs with a constant number of on-bits.
2. Maintain overlap properties between input and output where similar inputs should produce similar outputs.

The output of the SP for a given SDR will not necessarily resemble the layout of the input SDR, but will produce an output SDR from its active columns; that is, it will produce an SDR with the columns that are most active for semantically similar inputs.

The spatial pooler has the following parameters:

Column Count The number of columns in the pooler.

Potential Connections Percent The proportion of the input SDR's bits that each column will potentially connect to. These mappings are randomly distributed during initialisation

Number of Active Columns Per Inhibition Area The number of columns that can be active per inhibition area.

Connection Threshold The permanence value threshold to determine if a column is connected to a given input bit. During initialisation, these values are given a normal distribution about *Potential Connections Percent*, with approximately 50% being connected

Active Connection Increment The value to increase permanence values during learning

Active Connection Decrement The value to decrease permanence values by during learning.

The spatial pooler attempts to learn the spatial semantics, or fingerprint, of its SDR encoded inputs, it does this by generating another SDR consisting of its active columns. The SP is made up of a number of columns, where each column is *potentially* connected to a number (*Potential Connections Percent* × *Column Count*) of inputs. This emulates the behaviour of the brain where there may be thousands of potential inputs to each column and the strength of this connection can change over time (Hawkins et al. 2016). Each of these potential connections has an associated permanence score and a column is considered connected when its permanence value exceeds *Connection Threshold*.

For any given column to be active, the number of its potential connection permanences exceeding *Connection Threshold* (we call this *ConnectedCount*) must be in the top *Number of Active Columns Per Inhibition Area* of columns ranked by their *ConnectedCount*. When a column is activated for a given input, permanence scores for connections that map to an on-bit of the input SDR are incremented by *Active Connection Increment* and connections that map to an off-bit are decremented by *Active Connection Decrement*. This means that over time, particular columns will become sensitive to particular regions in the input, and that the regions the column is sensitive to may change depending on how the distribution changes over time. This enables the spatial pooler to perform online-learning and effectively adjust to changes in distribution.

Figure 3.2 provides a visual representation of an SDR input (left) and the columns of a spatial pooler (right):

- Blue cells in the input indicate ON bits
- White cells indicate OFF bits

- Each blue line represents a potential connection. This will considered connected if its current permanence value exceeds the *Connection Threshold*.
- Green dots on the input indicate an ON bit that is potentially connected to the highlighted SP column. These will be incremented by
- Grey dots on the input indicate an OFF bit that is potentially connected to the highlighted SP column

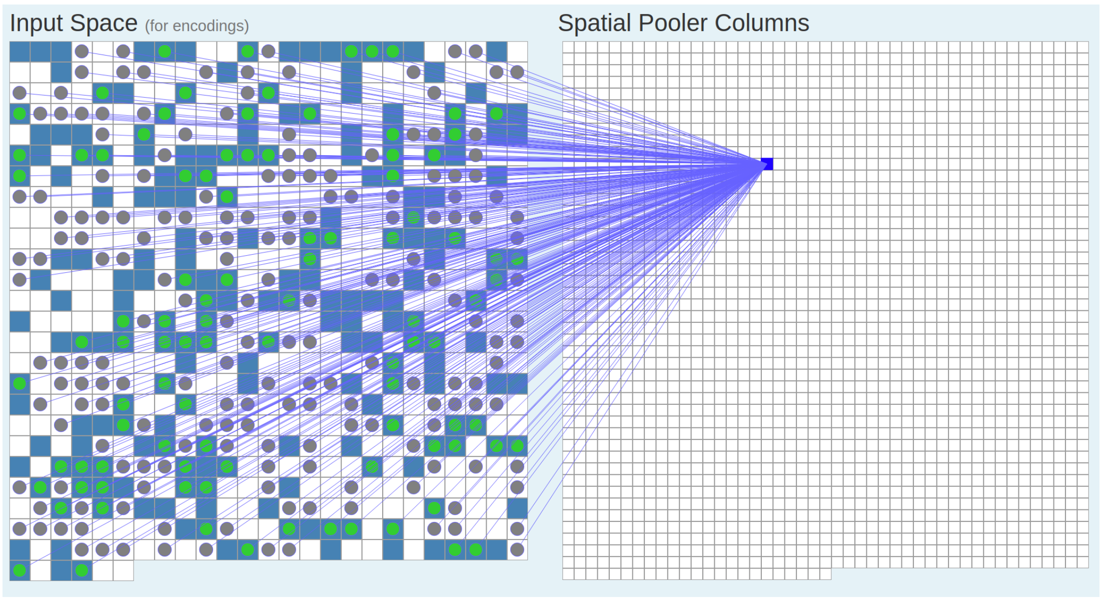


Figure 3.2: Connections from input SDR to a single Spatial Pooler Column [2]

In order to prevent only a minority of columns from ever being active, the SP implements boosting, which replicates the phenomenon of homeostatic regulation of neuronal excitability (Williams et al. 2013). In HTM, this means that as a particular column becomes more active, its permanence value will be artificially decreased to allow more inactive columns to become active. Conversely, increasingly inactive columns will have their permanence scores artificially increased to raise their chances of becoming active. This ensures that over time, all columns should become active and thus the SP becomes more robust and prevents a small subset of columns from dominating the output.

More formally, the SP algorithm can be described as follows (Hawkins et al. 2016):

1. An input SDR is created from some data
2. Initialisation: datastructures are allocated to allow the algorithm's execution before the first input is processed

[2]Graphic generated using the htm-school-viz tool suite by Taylor (2019)

3. For all columns, determine the active synapses from the current input based on those that exceed the connection threshold
4. Boosting: the count of each each column's active synapses is multiplied by a boosting factor
5. The columns with the highest number of active columns are now marked as active
6. Learning: for each active column, the synapses that exceed the permanence threshold are increased by *Active Connection Increment* and those below the threshold are decreased by *Active Connection Decrement*. Permanence values are clipped to the range $[0, 1]$. This process may bring some synapses above or below the permanence threshold
7. Outputs of the SP is the SDR of active columns.
8. Subsequent inputs repeat this process from step 3.

3.1.3 Temporal Memory

The aim of Temporal Memory (TM) is to learn the relationship between each input in the sequence it has seen (Hawkins et al. 2016). In a similar fashion to how the SP learns the spatial relationships between different encoded inputs by making connections between the input layer and the SP, the TM learns the sequential relationship between inputs by forming connections within its columns. It takes as input an SDR representing the active columns of the spatial pooler and performs two major tasks:

- Forming a representation of its inputs that captures the temporal context of previous inputs
- Making a prediction based on the current input in the context of its previous inputs

The parameters to the TM are:

Column Count the number of columns in the TM

Activation threshold the sum of proximal dendrite

Cells Per Column the number of cells in each column

Max segments per cell the maximum number of segments per cell

Max synapses per segment the maximum number of synapses per segment

Permanence Increment the amount to increase permanences by during learning

Permanence Decrement the amount to decrease permanences by during learning

The TM is made up of columns (the same number of column used by the SP), where each column has a number of cells that each receive the same subset of the feed-forward input via 'proximal dendrites'. Different columns are connected to different subsets of the feed-forward input. As an example, if the TM has 100 active columns each with 4 cells and only 1 cell per column is active at any given time, then there are 4^{100} different ways of representing the same input. This means that the TM can represent the same input in a large number of ways depending on the context, and these representations can be distinguished by checking their overlap score.

A cell can be in one of three states: inactive, active and predictive. A cell is active from a feed-forward input (from the previous layer), then it is termed 'active', if it is active due to inputs from neighbour cells in other columns, then it is in the 'predictive' state. Each cell has a potential connection to many cells in other columns called 'synapses' that have an associated permanence value representing the strength of the connection. Two cells are considered connected when their permanence value exceeds a specified threshold.

'Dendrite segments' are used to group the synapses between cells of which there are two types. Proximal dendrite segments connect feed-forward inputs to a cell and linearly summed, if this sum exceeds a threshold, the column is considered active. 'Distal dendrite segments' connect synapses between cells (and potentially to other cells within the same column) within the layer and are considered active when the sum of their synapse permanence values exceeds a threshold and the associated cell enters a predictive state.

In order to form a prediction on the current input in the context of previous inputs, the TM will take the active columns from the SP output. Hawkins et al. (2016) describe the TM algorithm for each input as:

1. For each of the active columns in the feed-forward input (the SDR received from the SP), activate the associated column in the TM.

2. For each active column, the predictive cells become active. If there are no predictive cells, all cells in the column become active. The set of active cells is thus the representation of the input in the context of prior input.

3. For each dendrite segment, count the number of connected synapses that correspond to currently active cells. If this count exceeds a threshold, the dendrite is considered active. Cells with active dendrites are put in the active state unless they are already active from feed-forward input. Cells

with no feed-forward input or active dendrites exceeding the threshold remain inactive. Columns with cells in the predictive state now form the prediction.

4. Whenever a dendrite segment becomes active, permanence values of all associated synapses are increased if the cell is active, otherwise it is decreased. These changes are marked as temporary.

5. When cells switch from inactive to active due to feed-forward input, each potential synapse connected to this cell has its temporary permanence changes status removed.

6. When a cell switches from active to inactive, temporary permanence changes are removed.

To this end, cells are activated based only on the feed-forward input, from which the TM will learn by strengthening connections to other cells that also activate with similar input. Additionally cells in a predictive state will become active only if they made a correct prediction, resulting in the TM learning and predicting sequences over time.

3.2 Applications of HTM

3.2.1 Making Predictions

Two classifiers are presented here that take as input the output SDR of the TM and make a prediction about a particular field k steps into the future:

Cortical Learning Algorithm the classifier used in the initial work in Chapter 4.

SDR Classifier a neural network based classifier used in the work in Chapter 5.

Neither algorithm is biologically inspired, but provide a useful tool for utilising the output of the TM, namely, predicting the value of a predicted field k steps into the future. They do this by learning a mapping from an SDR at time t (SDR_t) to a probability distribution over possible outputs k steps into the future.

3.2.1.1 Cortical Learning Algorithm Classifier

The CLA Classifier is not biologically inspired, but is a useful tool for interpreting the SDR output from the temporal memory and generating predictions. Essentially, it attempts to learn a function of an SDR at time t (SDR_t), such that it produces a probability distribution over the predicted field (PF), k steps into the future:

$$f(SDR_t) \to P(PF_{t+k})$$

The CLA Classifier takes the following parameters:

α The value used to compute the moving average. A lower α values give a longer memory

Steps The set of steps into the future that the classifier will learn and predict, eg. $\{1, 3, 7, 12\}$.

To do this, for each predicted step (k), the CLA Classifier maintains a mapping of:

$$f(SDR_{t-k}) \to PF_t$$

This mapping essentially stores a history of input SDRs it has seen, so, given an input, it can refer to the history and determine the probability distribution over the PF from a given input. It does this by:

- Storing two arrays H and A, with shape $N \times B$, where N is the number of bits in the SDR and B is the number of buckets on the PF as defined by the input encoding:

  H A histogram that stores the relative frequency of bucketed input values from when its corresponding SDR bit (n) is active. That is:

  $$H[n][b] = \frac{\text{times input was seen when } n \text{ was active}}{\text{times } n \text{ was active}}$$

  A A moving average of the input values, whose length is defined by α. When this array's corresponding SDR bit n is active with a given predicted field value v that falls into bucket b, the array is updated by:

  $$A[n][b] = ((1 - \alpha) \times A[n][b]) + \alpha \times v$$

  This ensures that when a bucket covers a range of values (ie. non-categorical values), the output is a prediction about the average value that fell into each bucket.

- For a given input SDR of length N with N' active bits, predictions are generated for each bucket (b) of the predicted field, at each timestep (k) by averaging the product of the associated histogram value and moving average table for each active bit:

$$P(PF_{t+k}) = \left\{ \frac{1}{N'} \sum_{SDR[n]=1} A[n][b] \times H[n][b] : b \in [1, B] \right\}$$

Thus we have a probability for each bucket of the predicted field, which may end up being very low for all buckets. We can use the bucket given the highest probability as our prediction, or not, depending on the context and the significance of the prediction. For example, the highest prediction may be for 100% engine load with a 0.1% probability, such a low probability would not necessitate the same response that a 95% probability would with the same load.

Another useful property of these predictions, is that they essentially form an ensemble, where on-bits associated prediction makes a small contribution to the final probability distribution.

3.2.1.2 SDR Classifier

The SDR classifier introduced by Cui et al. (2015), has the same goal as the CLA classifier, that is, it takes an input SDR from the TM and makes a prediction about a predicted field k steps into the future. It also does this by learning a the probability distribution function:

$$f(SDR_t) \rightarrow P(PF_{t+k})$$

The SDR classifier learns this using a single-layer feedforward neural network with a softmax output.

3.2.1.3 Neural Networks

An artificial neural network (ANN) is a type of model that takes as input a vector of data, and produces a vector output via finding a large number of weighted sums passed through an activation function. Typically these outputs are a classification or a prediction. For example, one could input a vector features about a fruit such as colour, size and mass, and the network would output a series of probabilities of target classifications, such as apple, banana, orange. Or, the network may take as input a continuous value, such as a traffic flow count, timestamp and produce a predicted value for the traffic flow k steps into the future (Zhang 2018).

The most simple ANN achieves these results by supervised learning, where the network is given inputs ($\mathbf{x}$) and expected outputs ($\mathbf{Y}$). Training the network involves repeatedly evaluating the network with input examples and then adjusting the weights so that its outputs are closer to the provided $\mathbf{Y}$.

It does this by storing 1 or more layers of cells (each with a weight value $w \in \mathbf{R}$), which are repeatedly adjusted until the output comes closer to the target classifications or predictions. It does this in 2 phases, the feedforward pass where outputs are produced, and the backward propagation of errors (commonly referred to as backpropagation) where layer weights are adjusted in order to reduce a loss function of the network's output.

A conceptual layout of an ANN is shown in Figure 3.3. It is important to note that initial values of these weights are normally set to 0, Although, it has been shown that intelligently selecting a weight distribution can improve performance, especially on networks with a large number of layers or when paired with an appropriate activation function (Glorot & Bengio 2010, Bengio 2012). For example, the 'Glorot uniform' (sometimes referred to as 'Xavier Uniform' [3]) initialisation uses a random uniform distribution in the range $[-limit, limit]$ where:

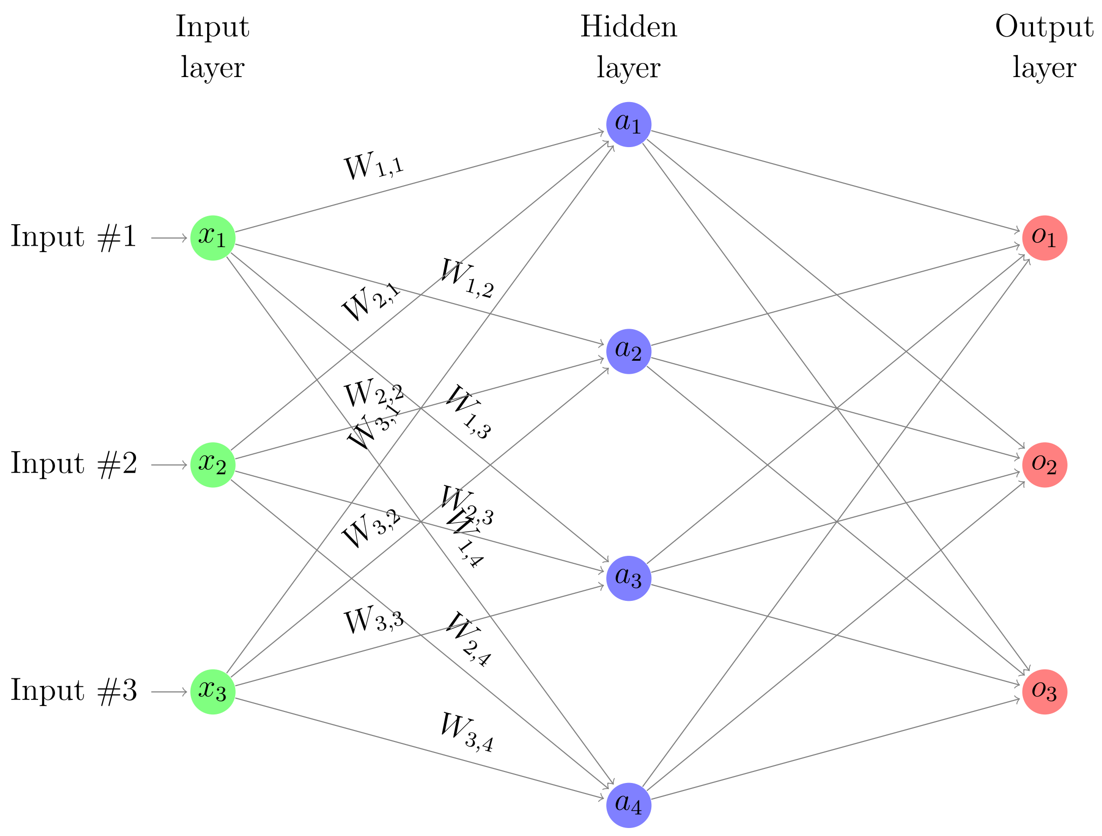


Figure 3.3: Layout of an ANN with a three inputs, a single hidden layer and 3 outputs

$$limit = \sqrt{\frac{6}{fanIn + fanOut}}$$

Where

[3]Named after its inventor, Xavier Glorot

$fanIn$ is the number of input units to the layer

$fanOut$ is the number of output units of the layer.

Two common tasks that ANNs can perform are regression and classification, each requiring an appropriate loss function:

Regression is the prediction of a continuous variable. For example the neural network may output a single value that is a prediction of a house price based off features of the house such as floor size, post code and number of bedrooms.

A typical loss function for regression tasks is mean squared error (MSE), given the true values y and the predicted values $\hat{y}$, $MSE = \frac{\sum_{t=1}^{n}(y_t - \hat{y_t})^2}{n}$.

Classification is the identification of inputs by labelling them with a class. For example, images can be labelled as fruit, animal or car, or given an audio sequence, predict which word is being spoken out of a vocabulary. Typically the output is a vector of probabilities that a particular input belongs to any of a set of predefined classes. Classification can also be used to make predictions about continuous variables by discretising the input into a number of buckets, and using the network's output predictions of the bucket with the highest probability as the prediction.

A typical loss function for classification is categorical cross entropy. Given the true distribution p and the predicted distribution q, this is calculated as $H(p,q) = -\sum_x p(x)\log(q(x))$. Al-Rfou et al. (2016) defines the cross entropy of 2 distributions over n categories, as a measure of the average number of bits needed to identify an event from a set of possibilities if the coding scheme used on a given probability distribution q rather than the true distribution p.

For example, if an input has the true distribution $p = (1.0, 0.0, 0.0)$, that is, the input only belongs to class 1, and a model gives a prediction of the distribution for the same input as $q = (0.4, 0.1, 0.5)$, then:

$$\begin{aligned} H(p,q) &= -\sum_x p(x)\log(q(x)) \\ &= -\left((1 \times \log(0.4)) + (0 \times \log(0.1)) + (0 \times \log(0.5))\right) \\ &\approx 0.916 \end{aligned}$$

Alternatively, if the predictions q are very close to the truth values, then the categorical cross entropy approaches zero. Given the same p, if q was $(0.98, 0.01, 0.01)$, then $H(p,q) \approx 0.02$. In the case of a neural network, it is the minimisation of this score for the network's output that the optimiser tries to achieve.

The loss function needs only to be differentiable for backpropagation to determine the gradients of weights of the network with respect to the error, so any such function will do.

Feedforward Pass The feedforward pass of a neural network is:

- The output of the input layer is the vector x of input values
- For each cell in each subsequent layer, multiply each input by a specific weight. That is, for each input to a cell, a corresponding weight is stored.
- For each cell, calculate the sum of the weighted inputs
- Apply an activation function f, to obtain the cell output. This must be a differentiable function so that the weights can be adjusted towards a minimum error during the backpropagation step
- For example, in Figure 3.3, the output of a_1 will be: $f(x_1W_{1,1} + x_2W_{2,1} + x_3W_{3,1})$
- The network's output is then the vector y of activations at the final layer.

Alternatively, for each layer n except the input layer (the output of the input layer is the input vector $\mathbf{x}$), the output of each layer can be expressed as the vector:

$$a_j = f_n\left(\sum_{i=1}^{N} W_{ij}x_i\right)$$

Where:

a_j is the activation of the j^{th} input unit to this layer

N is the number of units in this layer

W_{ij} is the weight for input i in unit j

x_i is the state of the i^{th} input

f_n is the activation function for layer n. Each layer can have any suitable activation function

In terms of computation, the weights of each layer can be stored as an array. Given a layer with I inputs and J units, this can be represented as a matrix L with I rows and J columns:

$$L = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1J} \\ W_{21} & W_{22} & & \\ \vdots & & \ddots & \\ W_{I1} & & & W_{IJ} \end{bmatrix}$$

Thus the output of a layer can be represented by a, the matrix multiplication of the vector of inputs x (with I features) with L. That is:

$$a = x \times L$$

This calculation can be extended to allow parallel processing of multiple input samples (often referred to as batches) by making x a matrix of I features as rows and K samples as columns. The output in this case becomes a matrix of K rows (representing the batch samples) and J columns (representing the features). Software libraries that implement neural networks will take advantage of parallel processing opportunities, especially GPUs which provide a large number of concurrent operations to be performed, to make these matrix multiplications as fast as possible and allow the training and implementation of very deep and very large networks viable (Oh & Jung 2004).

The final activation of the layer is the activation function f_n applied to each element in a, which may also be performed in parallel.

Backpropagation Pass As the weights of the network are initialised randomly (or to a single value such as 0 or 1), it is unlikely that the network will produce an accurate output. Before the discovery of backpropagation, network weights were set by hand in a tedious process that required significant domain knowledge and trial-and-error work. In order to gain the optimal set of weights for a network (a configuration that produces the lowest error score for the dataset we will eventually feed it), the weights must be modified in such a way that the error is eventually minimised. Choromanska et al. (2014) has shown that this minimum does not necessarily have to be the global minimum (which may incur overfitting where the network is unable to effectively predict inputs it has not seen) and that local minima have a similar function to the global minimum. There are several methods of doing this, the most common of which is backpropagation of errors (often referred to as ‘backprop’). Rumelhart et al. (1986) introduces the steps of backpropagation as:

- Given the output of the feedforward step for the last network layer: $\hat{\mathbf{y}}$, calculate the error using a loss function. The aim of the loss function is to provide a measure by which the error of the network is calculated and which backpropagation aims to minimise.

- Given a differentiable loss function E, backpropagation will attempt to minimise the error with respect to the neural network's weights. The first step is to calculate the $\frac{\partial E}{\partial y}$ for each output unit (y_j). Given the first derivative of E':

  $$E' = \frac{\partial E}{\partial y}$$

  Applying the chain rule to find $\frac{\partial E}{\partial x_j}$:

  $$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dx_j}$$

  The result is a function that describes how a change in x will affect the error. This means that we can find the gradient of the layer's weights with respect to the loss:

  $$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}} \\ &= \frac{\partial E}{\partial x_j} \cdot y_i \end{aligned}$$

  This function can now produce a function that describes the loss surface of a particular network and so the network's weights at each layer can be updated by:

  $$\Delta w = -\varepsilon \frac{\partial E}{\partial w}$$

  Where ε is a learning rate. In geometric terms, the network gradient $\frac{\partial E}{\partial w}$ is the direction and ε is the magnitude of the step toward a minima for the network's error. This use of a gradient and learning rate is referred to as Stochastic Gradient Descent (SGD). Modern variants of this process use more sophisticated updates incorporating normalisation, momentum and regularisation, the discussion of which are beyond the scope of this work.

3.2.1.4 SDR Classifier Neural Network

The SDR classifier uses a single layer ANN in order to learn the class probability of TM's SDR output. Given:

- $\mathbf{x}$ SDR output vector of TM, the activation state caused by an input SDR s_k of bucket C at step k .
- C_k the bucket index of the input to the HTM model at step k.

The SDR classifier attempts to learn the probability distribution of bucket indexes y_k given the TM's SDR output $\mathbf{x}$:

$$y_k = P(C_k|\mathbf{x}) = \frac{e^{a_k}}{\sum_{i=1}^{K} e^{a_i}}$$

It does this by using a single layer ANN with a softmax activation. Softmax activation is an activation function that squashes values into the range $(0, 1)$.

Thus, the output of the network will be a vector of probabilities in the range:

$$y_{k,n} = [0, 1)$$

Where $y_{k,n}$ is the probability that the input at step k belongs to bucket index n out of N possible buckets. Additionally, this vector will sum to 1:

$$\sum_{i=1}^{N} y_{k,n} = 1$$

And the predicted bucket is the index of the greatest value in y_k.

To summarise, the SDR classifier has the following architecture:

1. Inputs to the network are the output of the TM
2. The classifier has a single softmax layer with number of cells equal to the number of buckets
3. Outputs are a probability distribution of predicted buckets
4. Softmax layer weights are updated after each input by $\Delta w_{ij} = \alpha(y_j - z_j)x_i$ where:

   α is the learning rate

   w_{ij} is the connection weight from the i$^{\text{th}}$ input cell to the j$^{\text{th}}$ output cell

   z_j is the target distribution, which is a vector of size N filled with $\frac{1}{N}$

   y_j is the predicted distribution

To this end, the SDR classifier should learn which TM outputs correspond to a particular bucket index. Nominally, the network's final prediction at step k can be taken as the index of the maximum value from y_k.

3.2.2 Anomaly Detection

In addition to making future predictions, HTM can also be used to detect if the current input is anomalous (Ahmad & Purdy 2016). The aim of anomaly detection in the context of HTM and streaming data is to detect novel input sequences and previously seen sequences in novel contexts. For example, Figure 3.4 shows a stackplot of sensor readings for intersection 115 (see Figure 5.1) strategic input 2, from successive Thursdays in 2015. There is an anomalously high flow for one peak and 2 low peaks:

- The first on 24th September, was caused by a closure of the nearby Southern Expressway's southbound traffic (Josephine Lim 2015), causing that same traffic to be diverted through intersection 115.
- The following 2 low peaks occur on the 31st and 24th of December, caused because they are on or near public holidays and as such, fewer people travel on those days. Without this important context, it would appear that traffic flows on these days are anomalous, but they should not be, and ideally anomaly detection should indicate this if it was provided with the context of examples of those days from previous years and by providing the model with an input that indicates if a day is a holiday.

These first anomalous peak are the types of values that ideally should be marked as anomalous while the latter should not given appropriate context.

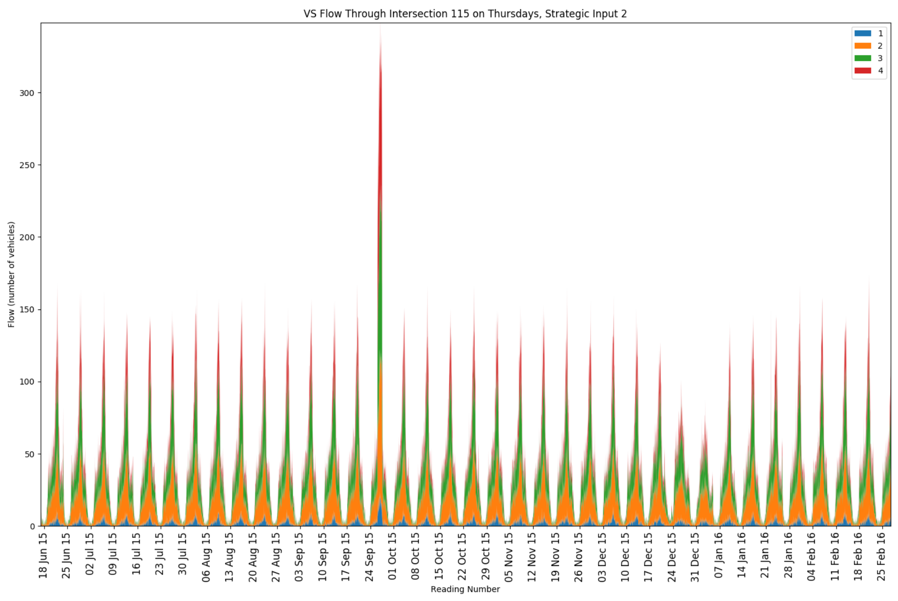


Figure 3.4: Thursday readings, with a particularly high peak

3.2.2.1 Algorithm

The anomaly detection algorithm in HTM takes as input the stream of data x_t, that is a vector of values at time t:

$$..., x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, ..., x_{t+n}$$

And produces two outputs:

Anomaly Score this is a raw value that simply measures the correctness of the previous prediction.

Anomaly Likelihood this is a percentage value of the likelihood that the current anomaly score indicates an anomaly in the context of previous inputs.

3.2.2.2 Anomaly Score

The anomaly score is the percentage of active SP columns that were not predicted by the TM. We use this because the aim of the TM is to predict the output of the SP, and working under the assumption that the TM has sufficiently learned the input sequences, should be a good indicator of whether or not the TM has encountered an anomaly.

$$anomalyScore = \frac{|A_t - (P_{t-1} \cap A_t)|}{|A_t|}$$

Where:

P_t Predicted columns at time t

A_t Active columns at time t

Alternatively put, the SP will produce an SDR at time t, we can get an anomaly score by calculating the percentage overlap between A_t and the predicted cells of the TM (P_{t-1}) *before* the TM has made learnt and inferred from the input at t. Thus, an anomaly score of 1 means none of the cells matched, and an anomaly score of 0 means all cells matched, ie. the TM made a perfect prediction about the state of the SP output at $t + 1$.

Assuming that the TM has had sufficient time to adequately learn the sequences of SDRs coming from the SP, *anomalyScore* can be used as a measure of anomaly. For this reason the anomaly score is usually ignored for some initial number of records, typically we can begin using the score after 500 records have been seen, but this is highly dependent on the context of the data. This is important because the TM can learn different contexts, so what may be anomalous on a weekday, may not be anomalous on a weekend, thus the anomaly score can be trusted as a robust measure regardless of context (so long as it has been learned).

3.2.2.3 Anomaly Likelihood

While Ahmad & Purdy (2016) describe anomaly score as an instantaneous measure of HTM's internal predictive performance, anomaly likelihood seeks to measure changes in predictability. That is, a value may be incorrectly predicted, but that does not make it unusual in context. For example, a sequence may be observed that has periodic changes and we would expect an increase in anomaly score when the changeover occurs but this is not actually anomalous because such changeovers are expected. Anomaly likelihood should indicate when previously unobserved sequences are encountered or those which are novel in an observed context.

Anomaly likelihood is calculated by (Ahmad & Purdy 2016):

- Keeping a window of the last W anomaly scores s_t
- Storing the mean (μ_t) and variance (σ_t^2) distribution over the anomaly score window
- A short term average ($\tilde{\mu_t}$) of W' anomaly scores is calculated where $W' \ll W$
- The final likelihood is calculated as the complement of the tail probability, also known as the Q function. The Q function is the probability that a normally distributed value is larger than x standard deviations. Given that Y is a normally distributed random variable with mean μ and variance σ^2, $X = \frac{Y-\mu}{\sigma}$, then $Q(x) = P(X > x)$ (Karagiannidis & Lioumpas 2007) where

$$\mu_t = \frac{\sum_{i=0}^{W-1} s_{t-i}}{W}$$
$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} s_{t-i}}{W-1}$$
$$\tilde{\mu_t} = \frac{\sum_{i=0}^{i=W'-1} s_{t-i}}{W'}$$
$$L_t = 1 - Q\left(\frac{\tilde{\mu_t} - \mu_t}{\sigma_t}\right)$$

An anomaly is reported if L_t exceeds some threshold (ϵ), that is when: $L_t \geq 1 - \epsilon$. Typically anomalies are only reported when the likelihood exceeds 0.99999 (equivalent to 4.5 standard deviations or a 1 in 147,160 occurrence).

3.3 Conclusion

In this chapter, the HTM background, processes and applications were described. HTM represents a family of algorithms that continuously learns to predict future sequences from input sequences through a biologically inspired model of the neocortex. This model takes a sparse encoding of input data, and outputs a learned representation of that data which can then be used by a classifier to make predictions about inputs in future steps, and to determine if the data is anomalous within the context of the observed input sequence.

HTM will be compared to other novel and existing methods for both prediction and anomaly detection in Chapter 4, Chapter 5 and Chapter 6.

Chapter 4

HTM and LSTM For Aggregated Traffic Prediction in CBD Locations

In this chapter, the focus of research will be to investigate the usage of 2 timeseries prediction techniques: HTM and LSTM in order to assess their effectiveness at predicting traffic flow in a variety of conditions using the VS data source. This chapter has been accepted for publication in *IEEE Transactions on Intelligent Transportation Systems* and has been cited multiple times , signifying is impact within the field. The description of HTM has been moved to Chapter 3 for a more comprehensive description. A more extensive explanation of LSTM models is also included.

4.1 Background

Traffic flow prediction is one of the most useful tools in ITS as it allows systems to act in a proactive way, facilitating the automatic adjustment of signal timings to optimise traffic flow through groups of intersections. Additionally, these predictions allow traffic controllers to provide accurate and timely traffic flow information for individuals, businesses and government (Tian & Pan 2015).

One of the main difficulties in predicting future arterial traffic flow is the non-linear nature of the raw data. Figure 4.1 shows that over the course of a day at intersection TS3044, there are clearly identifiable morning and afternoon peaks in traffic flow with drops during midday and throughout the evening. Even though these overall trends are apparent, individual data points show that there is no distinct pattern from one data point to the next aside from alternating local maxima and minima. Additionally, these patterns depend on the day of week, that is, in particular Fridays, Saturdays and Sundays (see Figure 4.2b) differ markedly to those from Monday to Thursday (see Figure 4.2a). Moreover, the

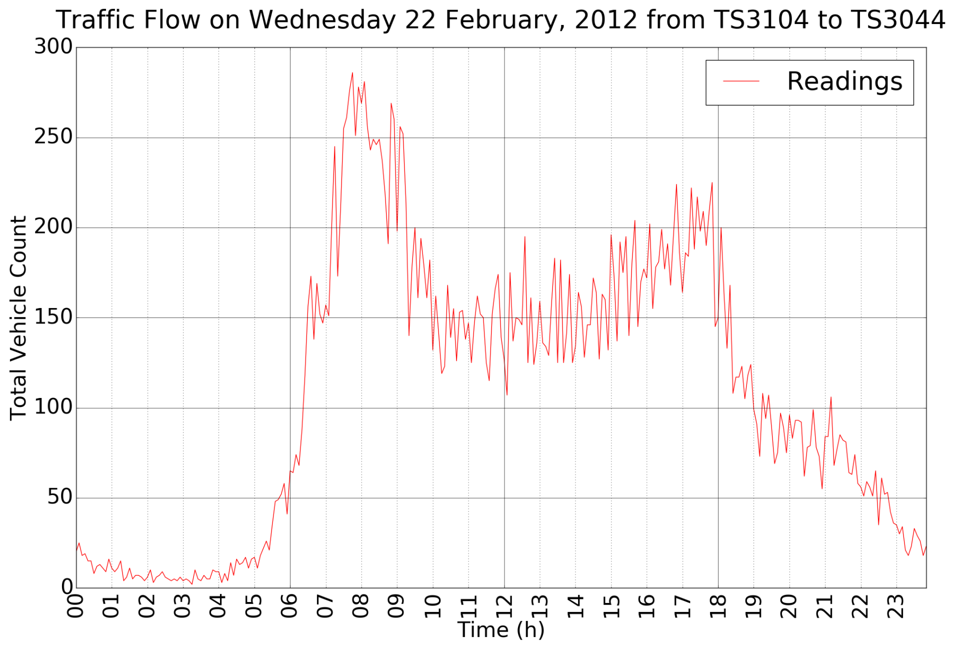
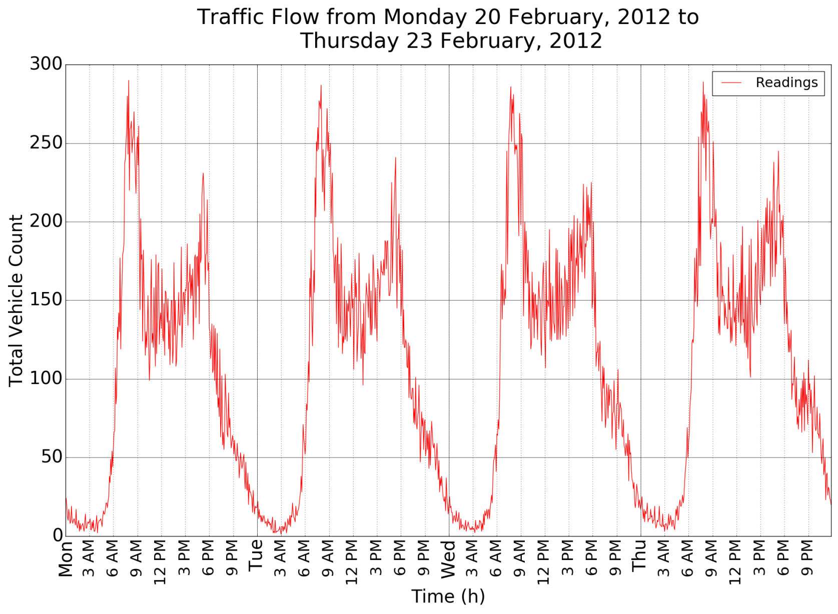
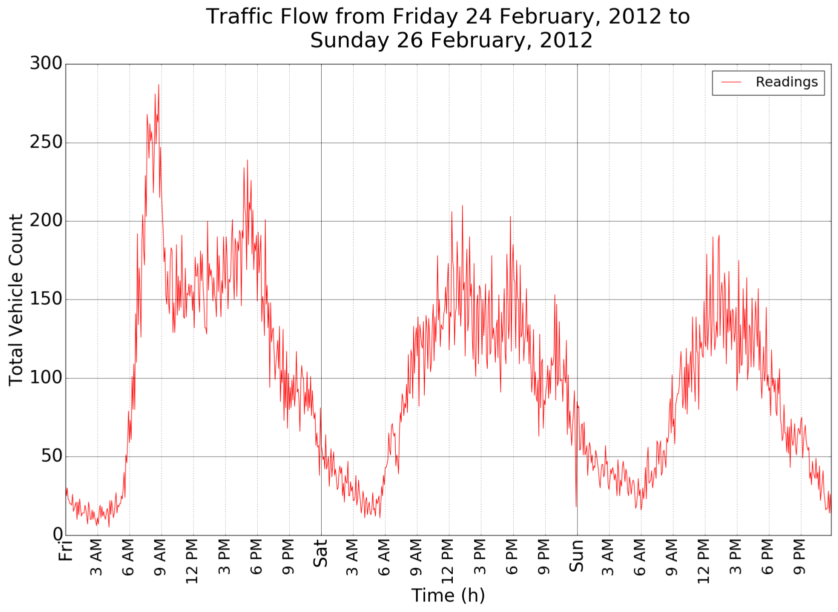


Figure 4.1: Typical Weekday Traffic Flow

overall distribution varies between weeks due to extended factors such as holidays, events and ongoing roadworks. Seasonal factors are also significant, for example, during summer it is reasonable to see increased traffic near metropolitan beaches and reduced traffic near schools. These observations concur with previous findings by Stathopoulos & Karlaftis (2001) which infer that short-term traffic prediction techniques cannot rely heavily on cycles within the data (such as daily, weekly or hourly) to accommodate the noisy nature of such spatiotemporal data.



(a) Monday to Thursday Traffic Flows



(b) Friday to Sunday Traffic Flows

Figure 4.2: Traffic Flow from TS3104 to TS3044

The nature of traffic flow as described necessitates the implementation of a predictive system that can not only deal with periodic distribution variations, but also sustained changes in distribution caused by external factors such as extended roadworks, accidents, road infrastructure changes, or hardware reconfiguration. To this end, an algorithm is required that can learn these patterns and, importantly, relearn when those patterns change. Online learning is one such method that allows models to adjust with each individual input which HTM performs. In this chapter, performance results of traffic prediction using HTM are compared to LSTM with and without online learning.

4.1.1 Related Work

This section examines the previous work into short-term traffic prediction. The majority of work into traffic flow prediction falls into three main categories (Lv et al. 2014):

- **Parametric** models that use time series analysis such as Kalman Filtering, Autoregressive Integrated Moving Average (ARIMA) and its derivatives,
- **Simulation** models that use traffic simulation tools to predict traffic flow, (see for example the work of Zhang (2016))
- **Non-parametric** models such as ANNs and *k*-nearest neighbours. These techniques are the main focus of our research.

Early use of neural networks for predicting traffic flow were first described by Smith & Demetsky (1994), where a network with 10 elements in a single hidden layer using backpropagation was used to learn vehicle counts over a 2 day period on a section of highway at 15 minute intervals. They compare results to historical average and ARIMA models, and show superior results and future potential for the use of neural networks prediction models with an RMSE of 2620, average absolute error of 144 (with peak volumes around 4,000 vehicles). Using such a small dataset is not sufficient to learn patterns or trends that occur over larger time scales at smaller intervals simply because the model does encounter them.

An analysis by Clark (2003) aimed to predict the traffic on London's M25 orbital motorway using multiple non-parametric regression over loop detector data at 1 minute intervals. The M25 uses variable speed signs in order to control traffic flow, thus they have a dataset combining speed limits, occupancy and traffic flow. Their model uses a *k*-nearest neighbour (KNN), that searches an historical database of observations for the *k*-nearest matches to the current state (occupancy and flow) and averages the result to make a prediction about the future traffic state and thus the optimal speed for that state. They tested this method on 3 weeks of data at 10 minute intervals, but due to the relatively small amount of training data used, it is unclear how the KNN method could be generalised in more varied traffic flow scenarios such as weekends or public holidays.

A database lookup method for imputing missing SCATS data is introduced by Vogiatzis et al. (2009), which essentially looks up different histories over each sensor and averages the result. This method could potentially be adapted to make predictions by treating the future readings as missing data, but no research is available regarding its performance in short term traffic prediction.

The majority of recent efforts in short-term traffic prediction involve the use of neural networks and is focussed around investigating the efficacy of their numerous variants and associated algorithms. A network is presented by Chan et al. (2012), where an exponential smoothing method is used to pre-process raw data and the network is optimised using the Levenberg-Marquardt algorithm (Lourakis 2005) for solving non-linear least squares problems. This scheme was evaluated at on/off ramps on the Mitchell Freeway in Perth, Australia for 6 separate weeks (Monday to Thursday) during AM peaks (7:30 - 9:30AM). Results showed that smoothing provided superior error rates to non-smoothed data.

Neural networks have seen a resurgence after the development of deep learning (LeCun et al. 2015), that is, networks with large numbers of hidden layers and appropriate techniques and algorithms that effectively train them, and the introduction of software libraries that employ the use of General Purpose Graphics Processing Unit (GPGPU) devices that make their implementation, training, and usage viable. One of the first such investigations into the use of these techniques for traffic prediction is by Lv et al. (2014), who examine the use of Stacked AutoEncoders (SAE) (Gang et al. 2015). Following this work, the use of deep belief networks Huang et al. (2014), Tan et al. (2016) and graph neural networks Shahsavari & Abbeel (2015) were also investigated.

A variant of recurrent neural networks term Long Short-Term Memory (LSTM) is first investigated by Tian & Pan (2015). They implement a single hidden layer with between 5 and 40 units and evaluate this model using the Californian Caltrans Performance Measurement System over a dataset for 6 freeways over 249 workdays in 2014 at 5 minute intervals. They compare the results of LSTM to random walk, support vector machines with radial basis function, single layer feed-forward neural network and SAEs. The results of this comparison showed that LSTM had superior mean absolute percentage error and root mean squared errors for all tested intervals. This chapter acknowledges these successes, and now seeks to build and extend on these results.

Another non-parametric method that does not use neural networks termed Fast Incremental Model Trees - Drift Detection (FIMT-DD) is introduced by Wibisono et al. (2016). FIMT-DD is a decision tree designed to model very large datasets that optimises the splits for each input attribute, such as link positions, link journey time, average link speed, link length and average link flow. This method is evaluated on data collected at 15 minute intervals over 5 years and 2500 individual sensors for motorways in the UK and evaluation shows that this method has a reduction in prediction percentage error, as data size increases, to around 15%.

In other work, Hamed et al. (1995) developed a time-series model for forecasting traffic volume in urban arterials using a Box-Jenkins auto-regressive integrated moving average (ARIMA) model, while Vlahogianni et al. (2005) used a genetic algorithm based optimisation strategy help model the traffic flow data in an urban arterial and Stathopoulos & Karlaftis (2003) employed a multivariate time-series state space model to provide traffic parameters such as traffic volumes, travel speeds and occupancies.

Driver behaviour varies between motorways, freeways and arterial roads (Ben-Akiva et al. 1984, Aghabayk et al. 2011) as does their connectivity and many other characteristics (Xie & Levinson 2007). Since the majority of research published around traffic prediction analyses freeway traffic and in limited time periods, this motivates the expansion of research into the arterial road domain with datasets covering multiple years of readings over the entire 24 hour period. The Adelaide traffic network in South Australia is a natural choice for such arterial road analysis as its road network mostly consists of arterial roads (with two expressways) (Norley 2011), whereas other cities have a much higher proportion of freeways and motorways to arterial roads.

To this end we also wish to verify that non-parametric metric methods (with their suitability demonstrated by Tan et al. (2016), Lv et al. (2014), Tian & Pan (2015), Shahsavari & Abbeel (2015)) such as HTM and LSTM can be transferred from the freeway and motorway domains to an arterial one. Additionally, previous studies have not addressed the effect of changes in traffic patterns and their effect on prediction quality. Furthermore, the above experiments often used limited datasets, limiting their evaluation periods, either to specific days of the week, hours of days or a few weeks in a year. This research seeks to evaluate the predictive efficacy of models that work around the clock over several years.

4.2 Long Short-Term Memory

Long Short-Term Memory is a type of recurrent cell in neural networks introduced by Hochreiter & Schmidhuber (1997) and is notable because it uses its output from previous timesteps as input as part of its evaluation. This recurrent input allows the network to learn long-term dependencies, allowing it to better predict sequence values especially where an output may be dependent on inputs from previous timesteps. These previous inputs may be from an arbitrary amount of time steps ago.

Additionally, the cells also have a ‘forget gate’, allowing it to reset its memory state. This structure makes sense, as humans attempt to read and understand a sentence, the previous words are kept in short term memory, allowing cognition of the sentence as new words are read in context with previous ones. Eventually the individual words and their ordering are forgotten but the meaning behind the sentence is understood (Olah 2015).

LSTM has been shown to produce state of the art results on a variety of tasks, especially time sequence prediction and classification problems including:

- Speech recognition (Tian et al. 2017)

- Weather prediction (Shi et al. 2015)
- Short term traffic flow prediction (Tian & Pan 2015)

4.2.1 Definition

The full definition of an LSTM cell is (Olah 2015, Jozefowicz et al. 2015):

$$
\begin{aligned}
i_t &= \sigma_1(W_{xi}x_t + W_{hi}h_{t-1} + b_i) && (4.1)\\
f_t &= \sigma_1(W_{xf}x_t + W_{hf}h_{t-1} + b_f) && (4.2)\\
o_t &= \sigma_1(W_{xo}x_t + W_{ho}h_{t-1} + b_o) && (4.3)\\
\tilde{c}_t &= \sigma_2(W_{x\tilde{c}}x_t + W_{h\tilde{c}}h_{t-1} + b_c) && (4.4)\\
c_t &= c_{t-1} \odot f_t + i_t \odot \tilde{c}_t && (4.5)\\
h_t &= \sigma_2(c_t) \odot o_t && (4.6)
\end{aligned}
$$

Where:

$\odot$ is the element-wise vector product operation

σ_1 is the recurrent activation function. Typically this is the sigmoid function: $\text{sigm}(x) = \frac{e^x}{e^x+1}$. This squashes its input into the range $[0, 1)$. In this work, the hard sigmoid is used, which is computationally faster linear approximation of the sigmoid: $\text{hardsigm}(x) = \max(0, \min(1, x \times 0.2 + 0.5))$

σ_2 is the cell's activation function. Typically this is the hyperbolic tangent (tanh)

W_* are the weight matrices. There are five sets of weight matrices stored: $W_i, W_f, W_o, \tilde{c}_t$.

b_* are the bias vectors, b_i, b_f, b_o, b_c.

f_t is the forget gate layer. This decides which values will not be passed onto the next layer. A low output here will cause the cell to forget its carry state. A high value will keep the carry state. It is important that the network learns to remember and forget the carry state through this variable.

i_t is the input gate layer. This decides which values will be passed onto the next layer.

c_t is the carry state, passed onto the next timestep

h_t is the memory state, passed onto the next timestep and is the output of the cell.

(h_t, c_t) the concatenation of h_t and c_t constitutes the hidden state which is the recurrent input for the next timestep.

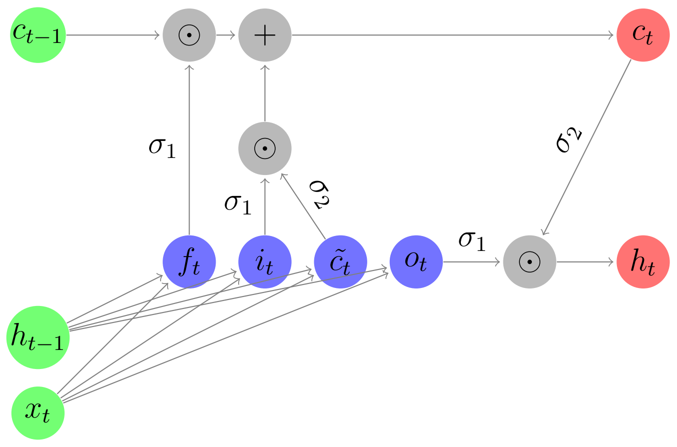


Figure 4.3: Layout of an LSTM cell where x_t is the input vector at timestep t. Activations are the activation functions at for each variable are shown on edges

See Figure 4.4 for plots of the activation functions. The layout of a LSTM cell is shown in Figure 4.3

As discussed in the 3.2.1.3, a layer in a neural network is composed of a series of LSTM cells, the outputs of which can be passed into other layer types including LSTM layers or simple densely connected layers.

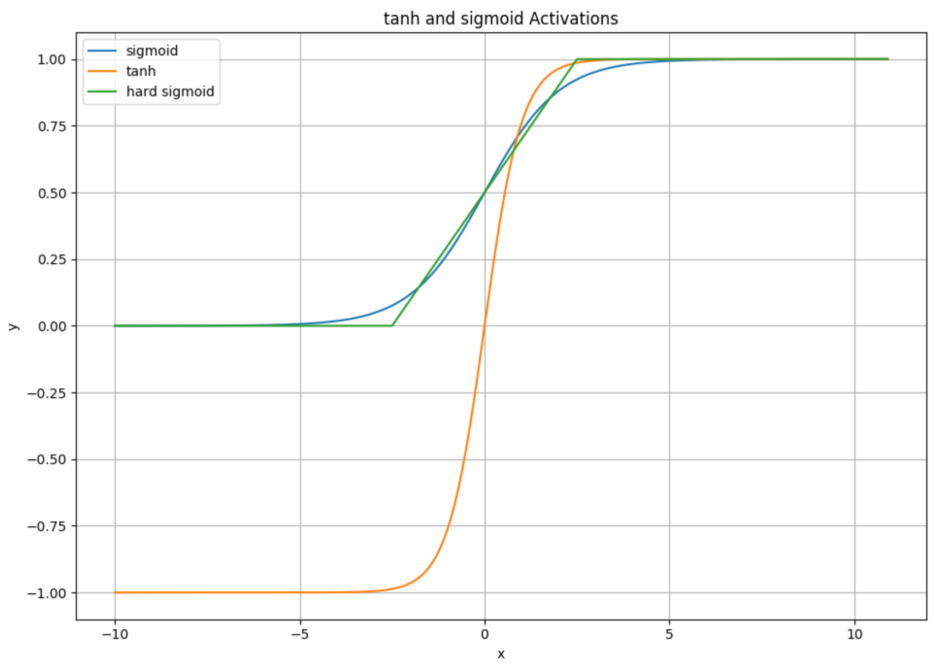


Figure 4.4: Sigmoid Activation

4.2.2 LSTM Architecture

In this work, the LSTM type models use a stacked model, where outputs of successive LSTM layers are passed onto the final dense layer. The last layer uses rectified linear unit ($f(x) = \max(0, x)$) activation and outputs a single floating point prediction for the traffic flow k steps into the future (for a value of k that the model has been trained on). It is possible to have multiple k from the same network by having multiple cells in the final layer, provided that the network is fed these values, but this configuration was not investigated and would be an avenue of future work.

4.3 Methodology

There are 3 main aims of this work:

1. Determine if LSTM can be applied to short term arterial traffic flow prediction given that it has been proven in the freeway and motorway domains.
2. Determine if HTM is suitable for short term traffic prediction, and
3. Evaluate HTM's performance in comparison to LSTM.

These tasks are solved by implementing the models and feeding them data from two intersections in the Adelaide central business district over a 2 year period.

4.3.1 Performance Measures

As these experiments are a regression analysis, the predicted values (y) are compared against the ground truth values (Y) and the following metrics calculated: Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE), Mean GEH[1]. GEH is a statistic commonly used to evaluate predictive traffic models (Feldman 2012) and is useful for evaluating predictions at small numbers (the dataset used contains times when there is zero flow, so any prediction near this, such as 1-10 would generate a large relative error, as with MAPE). MAPE and RMSE are recorded in order to make reasonable comparisons to previous work; the author argues that GEH is the most useful metric for evaluating traffic predictions. Calibration criteria provided by the South Australian Department for Transport, Energy and Infrastructure (DTEI 2010, p. 27) gives acceptable ranges for GEH when defining AIMSUN models as less than 5 for individual flows/movements and less than 4 for sum of all flows movements. These metrics are defined as:

[1] Named after its creator, Geoffrey E. Havers.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - Y_i|}{Y} \tag{4.7}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - Y_i)^2} \tag{4.8}$$

$$\text{MGEH} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{2(y_i - Y_i)^2}{y_i + Y_i}} \tag{4.9}$$

4.3.2 HTM

HTM was evaluated using the NuPIC implementation (Taylor et al. 2016). The model was fed the first 40% of the data to allow it to learn the patterns, followed by the remaining test data and statistics were collected regarding its predictions. The encoder parameters for the sensor count were decided by evaluating predictions for various bucket counts and using the one that produced the lowest GEH score[2]. Bucket width is determined by:

$$bucketWidth = \max\left(0.001, \frac{maxInput - minInput}{buckets}\right) \tag{4.10}$$

4.3.3 LSTM

The LSTM network architecture and parameters were determined by TPE hyperparameter optimisation (see section 5.2 for a full explanation) using hyperopt (Bergstra et al. 2013) and the best parameter set selected as the network with the lowest GEH score for the evaluation dataset. The search space and optimal parameters are described in Table C.1 and a full description of the optimisation algorithm is given in section 5.2.

The models were implemented using the Keras frontend (Chollet 2015) to Theano (Al-Rfou et al. 2016). Data was split into training and test data with 40% training, 60% evaluation. The inputs to the network are:

- Sum of lane counts
- Day of week
- Hour of day
- Minute of hour

[2]The source code repositories are available on github at https://github.com/JonnoFTW/traffic-prediction for LSTM and https://github.com/JonnoFTW/htm-models-adelaide/tree/master/engine for HTM.

The label is traffic flow k steps into the future, and it is this value that the network attempts to predict by minimisation of the Mean Squared Error (MSE) score using the Adam optimiser (Kingma & Ba 2014). Two models are evaluated, one with batch learning and another using online learning.

4.3.4 Batch Learning (LSTM-Batch)

The first method uses the traditional batch learning method, where the entire training set is split into batches, each batch is fed into the model, the error is calculated and backpropagation through time is used to update the weights in the network proportional to the error. This process is then repeated until the error score reaches a minimum. Whether or not this minimum is local or global depends on the type of optimiser and learning rate used (Jain et al. 2014). Once an acceptable minimum is found, the network is then used to process the testing set and statistics collected about the predictions.

4.3.5 Online Learning (LSTM-Online)

This method feeds data into a stateful LSTM network in small batches (or mini-batches), typically of 1-5 records, the error score is calculated and backpropagation through time (BPTT) is used to update the network weights (Jain et al. 2014). This training method allows the network to learn continually while also learning time based dependencies by remembering the state of from the previous batch LSTM cell's state.

During testing, the network is used to make a prediction from which the error score is calculated using the next value and the network weights are updated. Performance measures are then collected from these predictions. Additional testing was performed to see if resetting the carry state values improved performance.

4.4 Results

4.4.1 Datasets

The data used in these experiments is a time series containing a timestamp and the lane counts (recorded from stop line loop detectors) during the 5 minutes preceding the timestamp. The analysis here covers counts that correspond to traffic flow over 2 separate links in the period from 1 January, 2012 to 11 July, 2013. The first link examined is northbound to a 4-way intersection along a corridor (TS3044), the second is a southbound link into a 3-way inner-city intersection (TS3002); all counts are from the sensor readings at the end of the link. All models were fed the same input:

- Day of week
- Whether or not the day is on a weekend
- Time of day
- Sum of the sensor counts for link

If any individual sensor's count exceeded 200 vehicles, then the entire timestep is skipped so as to prevent the system from learning error values (this work is only concerned only with predictions about the state of traffic flow, and not the state of the sensor which may be in an error state).

In order to evaluate each model's effectiveness when distributions change, metrics are provided for the period 23 April, 2013 to 15 June, 2013 Table C.2. During this period, a single sensor's distribution (of the 3 sensors for the link) varied dramatically, see Figure 4.5, where a higher standard deviation indicates a significant variance in traffic flows throughout a given day.

- Up until 1 May, 2013, the morning peak peaked at around 30 vehicles at 9AM
- Between 1-7 May, it peaked at around 20 vehicles at 9AM
- Between 7 May and 8 June, 2013, it began collecting at distributions similar to a tram sensor; peaking at 4 vehicles per interval, with a significantly higher than normal error count
- After 8 June, it returned to peaking at around 20 vehicles.

Each model was evaluated during this period at $k = 1$.

4.4.2 Analysis

The results show that HTM, LSTM-Batch and LSTM-Online can indeed be applied to the domain of short term arterial traffic prediction as they give RMSE scores superior to other state of the art usage in the freeway domain Tian & Pan (2015), Wibisono et al. (2016). The MAPE scores provided by all models used here are worse than those in previous works, but this is likely due to the increased amount of noise in arterial traffic flows compared to freeways and motorways.

Results from both datasets, Table 4.1 and Table 4.2 show that HTM suffers from a loss in prediction quality as k increases while LSTM does not, this is due to the fact that each LSTM model is trained for a specific k whereas HTM does not.

A subset of predictions for the varied distribution dataset are listed in Figure 4.7 for HTM, Figure 4.8 for LSTM-Batch, and Figure 4.9 for LSTM-Online.

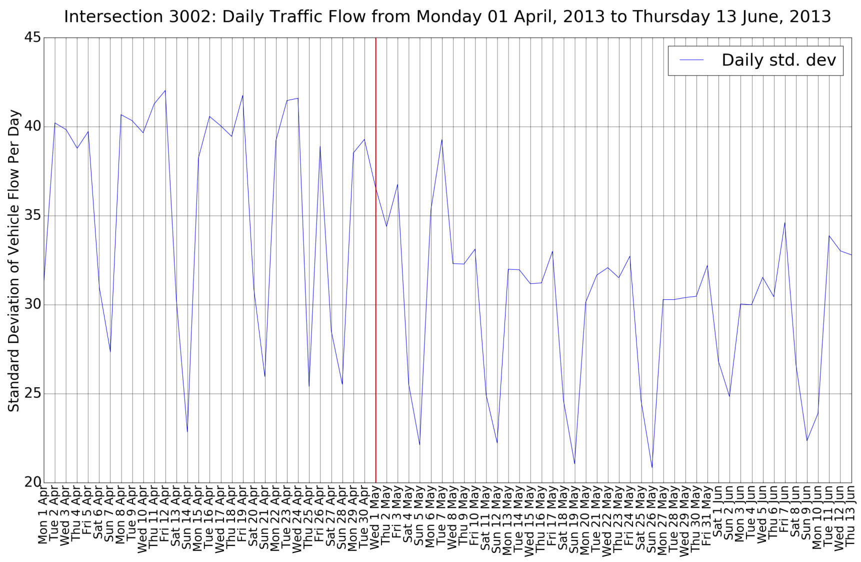


Figure 4.5: Standard deviation of traffic 5 minute vehicle flow readings per day, showing that there is both a change in traffic flow variability throughout the day, and over time

It can be observed here that both HTM and LSTM-Online adjusted to the change in volume. Whether or not this behaviour would be useful in this case is debatable as it would be highly dependent on the cause of the change, for example, if a single detector were to be in an error state, adapting to the new sensor counts would cause the model to make predictions about the remaining sensors and not the actual number of vehicles on the road. Nevertheless, it can be assumed that the large MAPE errors for LSTM-Batch on TS3002 are caused by its inability to adapt to change in the absence of the online-learning seen in the other two models, see Figure 4.8.

A summary of results for each intersection tested are provided in Table 4.1 and Table 4.2. Results for TS3002 specifically during the period of varied distribution are in Table C.2.

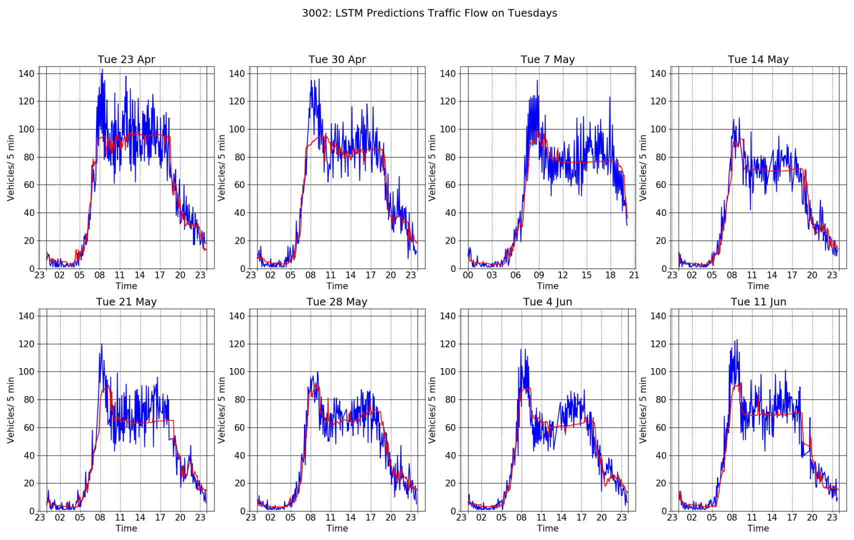


Figure 4.6: LSTM Predictions for successive Tuesdays

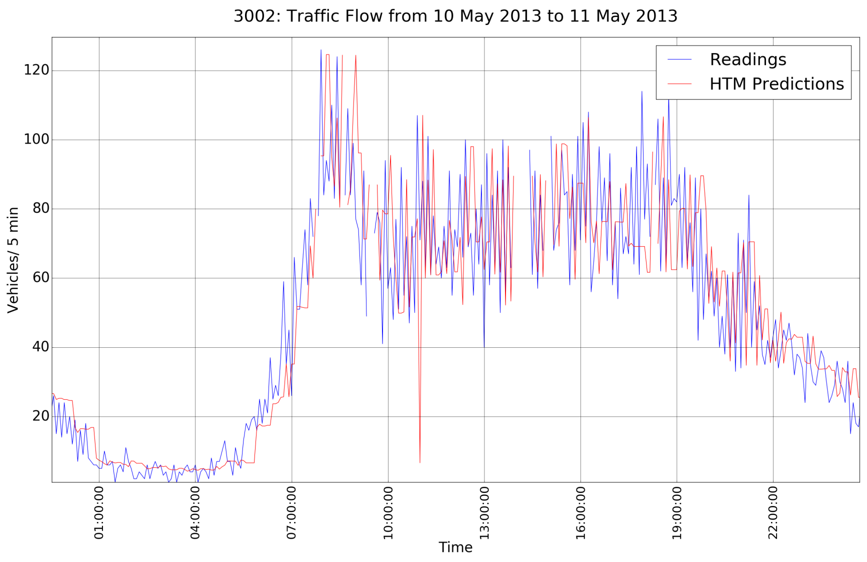


Figure 4.7: Zoomed Area for Traffic Flow Distribution and HTM Predictions

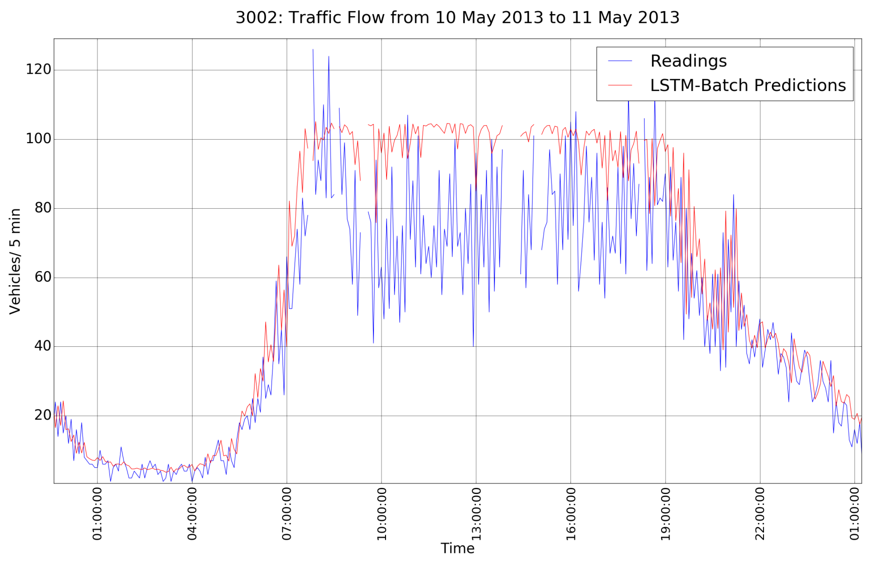


Figure 4.8: Zoomed Area for Traffic Flow Distribution and LSTM Batch Predictions

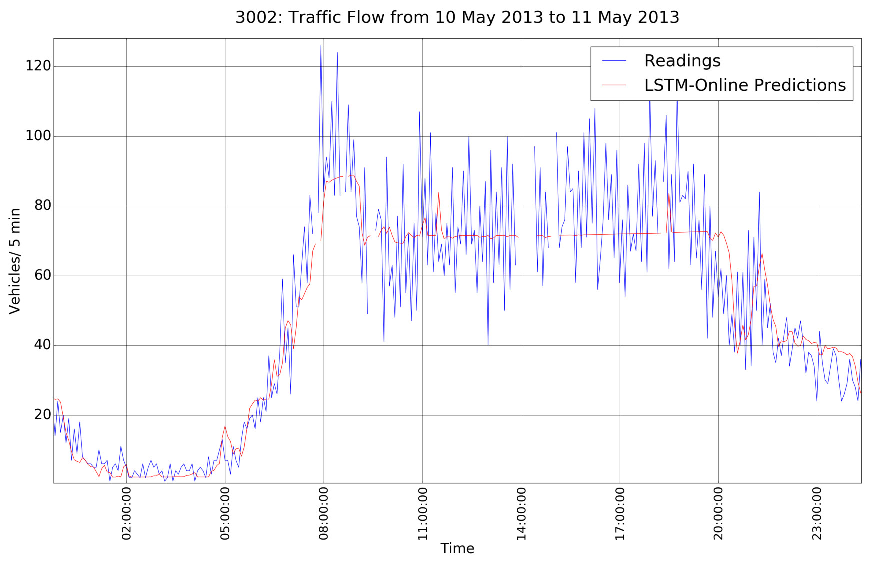


Figure 4.9: Zoomed Area for Traffic Flow Distribution and LSTM-Online Predictions

Table 4.1: Summary of Results for TS3044

Time Step (k)	Model	HTM			LSTM-Batch			LSTM-Online		
	Metric	GEH	RMSE	MAPE	GEH	RMSE	MAPE	GEH	RMSE	MAPE
5 Minute		1.78483	24.3256	25.6438	**1.4327**	19.1937	**20.9013**	1.6701	**15.5476**	27.9095
15 Minute		1.92271	26.1495	28.1514	**1.4552**	19.6388	**21.6877**	1.6324	**15.3397**	25.3137
30 Minute		2.13217	28.6643	31.8891	**1.4688**	19.6402	**22.2586**	1.7231	**16.1951**	27.1289
45 Minute		2.27641	30.1221	34.9188	**1.5125**	20.4665	**22.115**	1.8050	**16.9344**	29.0042
60 Minute		2.37946	31.4697	37.2002	**1.58235**	21.4208	**23.2616**	1.91873	**18.009**	31.3967

Table 4.2: Summary of Results for TS3002

Time Step (k)	Model	HTM			LSTM-Batch			LSTM-Online		
	Metric	GEH	RMSE	MAPE	GEH	RMSE	MAPE	GEH	RMSE	MAPE
5 Minute		**1.36306**	12.4738	**33.1574**	1.59367	15.8388	40.2894	1.5299	**10.525**	39.68655
15 Minute		**1.50805**	13.8317	**36.7716**	1.57015	15.6433	39.5736	1.52929	**10.5314**	38.955
30 Minute		1.80306	16.7724	43.7712	**1.41763**	13.1827	**38.3119**	1.45773	**9.75738**	39.5527
45 Minute		1.97191	18.248	49.9565	**1.57309**	15.6521	41.2436	1.57453	**10.8052**	**40.9037**
60 Minute		2.24488	20.9259	57.6634	**1.43799**	13.5727	**40.4709**	1.5429	**10.3995**	41.4226

4.5 Conclusion

This chapter presents the findings of an investigation into the suitability of HTM for short-term traffic prediction in an arterial road network and compare it to LSTM models. The efficacy of HTM to learn and predict short term arterial traffic flows is clearly demonstrate, competitive performance with another state of the art method (LSTM), and novel given the state of literature on arterial and freeway traffic prediction. Furthermore, findings show that LSTM can be transferred from the highway and motorway domains into the arterial road domain and that improved performance can be found when LSTM is used in an online learning fashion. It was found that LSTM gives improved performance over HTM at the cost of training and running 5 separate LSTM models per link; although HTM can adequately predict traffic flows based on its GEH scores. It was also demonstrated that both HTM and LSTM-Online have the benefit of adjusting to changes in traffic flow distributions. Despite HTM's decreased performance at increasing k, it is suitable as a tool for short term traffic prediction on arterial roads.

Avenues for future work around predicting short term traffic flow include:

1. The optimisation of HTM parameters, especially experimentation around encoder parameters and input feature selection.

2. Investigating parameters for a single LSTM network that outputs a prediction with multiple outputs, one for each desired time step, rather than separate models for each timestep.

3. Investigating a gated recurrent unit (a simpler variant of LSTM) model as described by Cho et al. (2014).

4. Investigating an LSTM model that uses a convolutional LSTM cell as described by Shi et al. (2015).

5. Experimenting with feeding additional parameters to the models, including the flow counts of upstream, downstream or other nearby intersections, public holiday status and the week of year. The fields would ideally be included in the search space of additional hyperparameter optimisations.

6. Investigating how anomalous traffic flow can be detected from the output of predictive models and then used to infer the presence of incidents.

Points 1 and 4 are investigated in Chapter 5 while 6 is investigated in Chapter 6.

Chapter 5

Predicting Next Phase and Aggregated Traffic Flows in Urban Areas

5.1 Introduction

Motivated by initial successes in predicting short term traffic flows on 2 intersections in the Adelaide CBD with HTM and LSTM based models, research effort is now turned to:

- Applying HTM and LSTM-Online techniques to intersections in other arterial areas (ie. those areas not in the CBD).
- Applying Seasonal Autoregressive Integrated Moving Average (SARIMA) models to the traffic prediction problem.
- Applying a Markov Model approach to traffic prediction.
- Improving hyperparameter optimisation search space and methodology, additionally searching over selecting input features in HTM and LSTM models. That is, models were trialled with and without certain inputs.
- Verifying that these methods work on both VS (5 minute aggregated dataset) and SM (phase level dataset) with a 1 step prediction task at an urban intersection.
- Making a comparison of these models and their transferability between VS and SM datasets.

Optimisation tasks were performed with a much wider parameter search space and on data that is free from erroneous flow values due to sensor flickering but also experiences anomalous traffic flows and changes in distribution. In this chapter, additional prediction tasks using HTM, LSTM-Online and SARIMA are evaluated in their effectiveness on traffic flow prediction and the suitability of TPE-based hyperparameter optimisation is demonstrated for HTM and LSTM.

Intersection 115 is examined (shown in Figure 5.1). It was selected because of its consistent dataset containing a minimal number of error values and no flickering across a two year period. Observing the VS dataset generally in Figure 5.3, at a daily level, clear weekly cycles can be observed:

- Weekdays Tuesdays through to Thursday have similar flows
- Fridays, Saturdays and Sundays all have distinct total flows

There are also several outliers in this dataset, for example:

- The last weeks of December and first week of January have significantly lower total flow. Given this context, these weeks are outliers but not anomalous.
- A large spike on 24 September, 2015 was caused by an accident elsewhere in the network that caused significantly more traffic to be routed through this intersection.

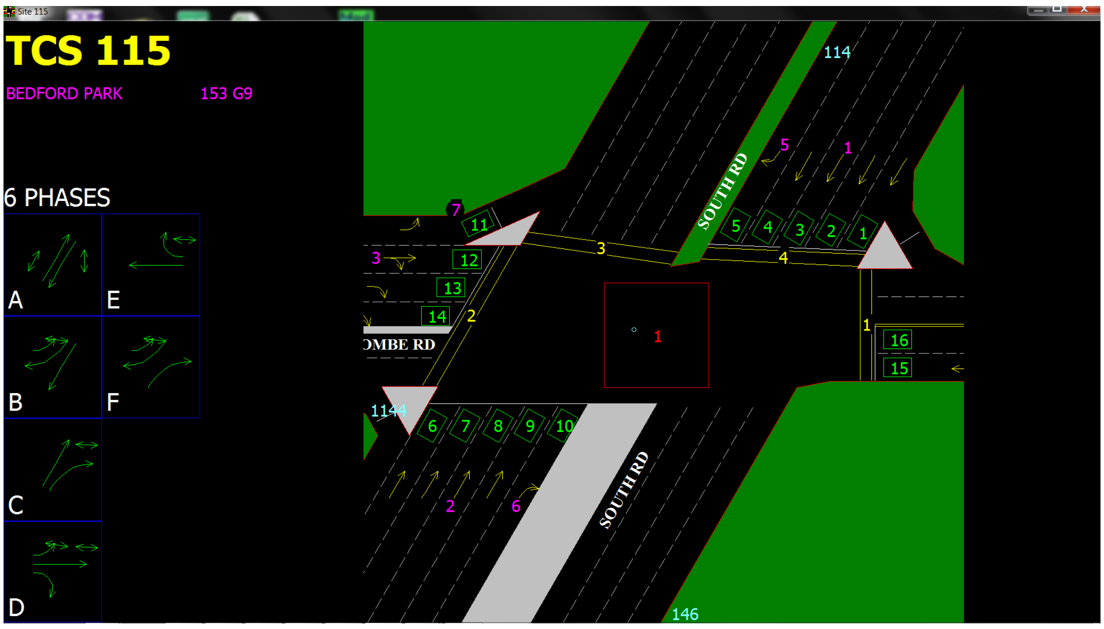


Figure 5.1: SCATS Diagram of Intersection 115

Due to the random and irregular nature of short-term traffic flow (as described in Chapter 4 and by Stathopoulos & Karlaftis (2001)), it is imperative that models be devised that can avoid concentrating on the mean of previous values as ARIMA models are likely to do (Angeline et al. 1994). This becomes apparent when examining the sign of the change between each volume count, for VS data, 46.25% of changes are positive, 46.61% changes are negative and 7.13% have no change.

The method of automatically choosing the best model hyperparameters and model configuration has previously been investigated by Vlahogianni et al. (2005). They use evolutionary hyperparameter optimisation techniques to determine ideal hyperparameters and network configuration for a Multilayer Perceptron (MLP) model. In the absence of memory units (as seen in more modern LSTM models, (Lv et al. 2014)), and in order to capture the temporal nature of the data, the input to their model included volumes for n previous steps for the target, upstream and downstream sensors. This particular model used a 3 minute aggregated traffic volume count dataset and evaluated 2 models, a univariate model that included the last 3 readings from the sensor and a multivariate model that also included the last 3 readings from the nearby sensors. Their results showed that genetic algorithms (and by extension, meta-learning algorithms) can improve the performance of neural network models for short term traffic prediction and find their optimal structure. As such, work in this chapter is driven by previous successes (in Chapter 4 and in previous work) to find the optimal model structure for various types of models to the short term arterial traffic prediction problem in Adelaide.

5.2 Hyperparameter Optimisation

Hyperparameters are the factors used when training a network (as opposed to the weights of the networks themselves), such as learning rate, dropout, initialisation scheme, optimiser algorithm, batch size, layer count, layer type, layer shape, activation function and input features. Appropriate hyperparameters vary depending on task and may not be immediately obvious and as such the task of hyperparameter optimisation seeks to automatically select the best hyperparameters for a given model architecture and task. Moreover, the search space can be expanded by finding the best parameters on a per layer basis, for example, layer shape and learning rate can be optimised for each separate layer.

The selection of such parameters can significantly improve a model's predictive performance (in addition to conventional parameter learning via methods such as stochastic gradient descent) and as such, a trade-off exists between the amount of computing time spent learning the weights in a particular network, and time spent searching the experimenter's defined search space. To this end, algorithms have been devised to find the optimal set of hyperparameters for a specified fitness function f, where f returns a fitness score for a particular model using the provided hyperparameters. Such fitness functions can encapsulate a single value (such as accuracy or root mean squared error), or a combination of values (such as accuracy, parameter count, and execution time).

The search space must be devised by the experimenter and typically involves specifying a probability distribution and range (along with mean and standard deviation where applicable) for each hyperparameter, such as uniform, Gaussian, log-normal and categorical. These distributions may also be quantised where discrete outputs are desired, as opposed to floating point values. Because the search spaces may be very large, contain many local minima and that the execution of models with any particular selection from the search space may take a several days, the development of algorithms that most efficiently find a global minimum (or close approximation) are of great interest. The benefits of such techniques are:

1. Experimenters do not need to spend what is often inordinate amounts of time in the process of selecting parameters and network configurations, training the network, evaluating results and repeating. Significant research work has gone into finding such optimal network configurations by hand, for example the task of image classification has received significant attention (Krizhevsky et al. 2012, He et al. 2016, Simonyan & Zisserman 2014, Long et al. 2015) after the publication of image datasets containing millions of labelled samples such as CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton 2009), and ImageNet (Deng et al. 2009). Automating this task frees a researcher's time for other work.

2. Model execution can be performed concurrently, scalable with the number of machines available for use.

3. The resultant models are likely to perform better for a specific problem and may identify novel configurations applicable to other tasks.

One drawback of such methods, is that the search space may be quite large and require more computational time than researchers have access to or can readily afford (although this does provide additional motivation to find efficient hyperparameter optimisation methods).

In general hyperparameter optimisation takes a fitness function (f, in this work f is the RMSE of model predictions on the last 60% of readings), search space (M), parameter selection function (S, which takes the historical evaluation results H and M), a maximum trial count (T) and returns a history of function evaluations over the search space (see Algorithm 5). The challenge lies in selecting S such that it produces a set of parameters that result in a minimum being found over f. The optimal set of evaluated hyperparameters is then the minimum from

H with the lowest fitness function result.

Algorithm 5: Hyperparameter Optimisation

```
Input: $T$, $f$, $S$, $M$
Output: $H$
1 $H \leftarrow []$;
2 for $t = 1$ to $T$ do
3     $x* \leftarrow S(H, M)$ ;
4     H.append($[x*, f(x*)]$);
5     /* History has parameters and their resultant score     */
6 end
7 return $H$
```

The simplest such method is grid search, where S iterates over all combinations of specified parameters. Such a method would require T to be very high ($T = \inf$ in the case of an exhaustive search) so as to evaluate all possible combinations and subsequently would require significantly more models to be evaluated than a method that takes a more considered approach. Another method is random-search, where S simply draws candidate parameter values randomly from their distributions.

A similar type of random search is simulated annealing (SA), as proposed by Kirkpatrick et al. (1983) and has shown to be effective at finding global minima in non-linear search spaces with numerous local minima (Lewis 2007). SA is a global optimisation technique that emulates the controlled cooling process of metal, where the crystal structure of a material experiences greater localised changes at high temperatures and eventually settles at low temperature. The algorithm works by:

1. Picking a random initial state S
2. Set $d \leftarrow f(S)$
3. Selecting a starting temperature T
4. For a given number of trials:
    (a) Pick a random neighbour state S_{new}
    (b) Calculate $d_{new} \leftarrow f(s)$
    (c) If $e^{\frac{-(d-d_{new})}{T}} >$ random uniform value between 0 and 1 then $S \leftarrow S_{new}$
    (d) Decrease T using cooling scheme
5. Return S

In this way, SA can escape local minima randomly by selecting locally suboptimal states, which occurs less as the system cools. Random restarts may also be used in to attempt to find the global minimum from a variety of starting states. Performance may also be dependent on the cooling scheme, which is typically logarithmic, which makes large changes more likely during early trials, and small changes more likely during later trials.

Tree-structured Parzen Estimators (TPE) is an optimisation technique proposed by Bergstra et al. (2011) as an algorithm for optimizing the hyper-parameters of neural networks and deep belief networks. Additionally, they also show that TPE is capable of training deep belief networks where random-search is not able to and achieve superior results over manual tuning. TPE works by substituting the specified distributions by:

- Uniform with truncated Gaussian mixture
- Log-Uniform with exponential truncated Gaussian mixture
- Categorical with re-weighted categorical

Bergstra et al. (2011) says that, given different evaluations ($\mathbf{x} = \{x^1, x^2, \ldots, x^k\}$), these substitutions represent a learning algorithm that produces densities over the original search space, producing two densities $p(x|y)$ (x is a particular parameter selection, y is the fitness score) where:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* l \\ g(x) & \text{if } y \geq y^* \end{cases} \tag{5.1}$$

Where $l(x)$ is the density of observations $\{x^i\}$ such that $f(x^i) < y^*$ and $g(x)$ is the density of remaining observations. TPE selects $y*$ to be a quantile γ such that $p(y < y^*) = \gamma$. Thus after each evaluation, new substitute distributions are selected in the search space so as to maximise the likelihood under $l(x)$ and minimise the probability under $g(x)$. Neighbour distributions over an interval (a, b) are selected over the observations in $\mathbf{x}$, so gradually as more models are evaluated, the density distribution of each parameter is likely to centre at an optimum for f.

One other method for optimising hyperparameters is the “genetic algorithm” technique, demonstrated by Real et al. (2017), with results comparable to state-of-the-art human designed models. This process uses a population based optimisation scheme to automatically determine the optimal neural network structure for an image classification task. It works by creating a population of very simple neural network models (where models are marked as *alive* or *dead*), a worker program selects two living models randomly, compares their fitness after training for a set number of iterations (although this is not enough to fully train the network under a single model training scheme), kills the weaker one and the other becomes a parent. The parent model will then have a child with a mutation applied, its fitness is evaluated and finally the child is returned to the population as *alive*, see Figure 5.2.[1]. Each mutation may be one of several different operations: learning rate change, do nothing (ie. continue training), reset weights, add an untrained convolutional layer at a random position, remove random convolutional layer, modify a random convolutional layer's filter size, add skip (where the outputs of one random layer are passed to a random deeper layer) and remove a random skip.

The end result after a specified number of models have been evaluated, or the fitness score does not increase is a fully trained model with optimal structure. This method can easily be scaled to an arbitrary number of worker processes to speed up the exploration of the search space and the discovery of the most efficient model structure for a given problem. As the authors mention, this is a drawback of the method, the significant computation cost of evaluating so many, often highly complex and deep models requires many hundreds of machines in order to effectively explore the search space. With this method they were able to achieve a 94.6% test accuracy with the need for post-processing, compared to 96.4% of the state of the art model on the same 10 class image classification task described by Zoph & Le (2016).

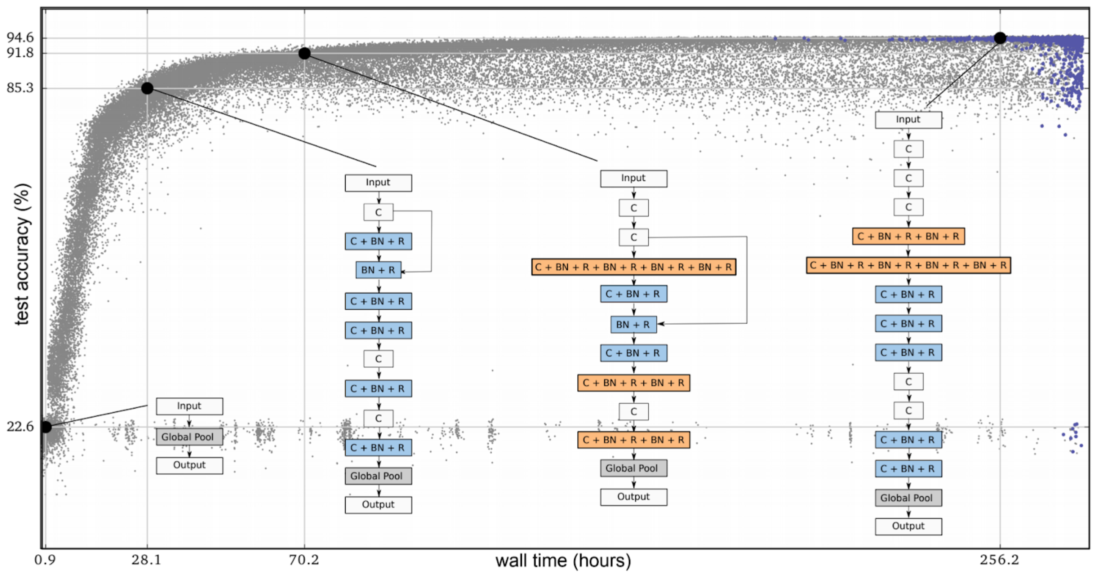


Figure 5.2: Test Accuracy over time where grey models are dead and blue models are alive. Model complexity is increasing over time along with the highest achieved fitness score. It is purely by chance that some very poorly performing models are still alive (Real et al. 2017)

[1]This method of selection where a loser is eliminated from the population and the winner lives is referred to as "tournament selection" (Goldberg & Deb 1991) and is by no means the only such selection scheme, their discussion is beyond the scope of this work

5.3 Methodology

In this research, a greater number of models (selected primarily due to their ability to perform online learning, along with some baseline models) were examined for the traffic prediction problem from simple to more complex:

- HTM
- LSTM-Online
- SARIMA
- Markov Model
- A simple n shift model: x_{t-n}, where the prediction is the n$^{\text{th}}$ previously observed value
- Rolling mean of the previous n values. Where the prediction for time t is $\frac{1}{n}\sum_{1}^{n} x_{t-i}$.

In order to determine optimal parameters for each model to the VS and SM 1-step prediction problem, hyperparameter optimisation via TPE was performed for HTM, LSTM and SARIMA. Due to the small parameter space, grid search was used for x_{t-n}, rolling average and Markov models. The models are all trained on the first 60% of data according to their timestamp (ie. values where fed to each model in the order they occurred, without shuffling). All models are evaluated based on predictions from the last 40% of flow readings in order to give learning models time to adequately learn the distribution of the data. As in previous work, RMSE is used to compare model performance.

In typical supervised machine learning tasks, the data are separated into three non-overlapping groups:

Training the data used to train the model and by which model parameters are adjusted according to the loss function. An epoch is when the model has trained on the entire training set once. Training typically stops after a predefined number of epochs have run, the validation score stops improving, or some other early stopping condition is met [2].

Validation the data used to evaluate the model's performance at the end of each epoch. The validation score is calculated in the same way as the testing score, and used to ensure that the model is not overfitting (where the model simply memorises the training set and fails to adequately generalise on the validation and testing sets).

[2]A full discussion of supervised learning training schemes is beyond the scope of this work

Testing the data used after all epochs have run to give a final performance measurement.

A validation set was not used in the present study because subsequent epochs of training did not increase predictive performance on the testing set, while subsequently increasing computation time and running the risk of overfitting.

In this section, the particulars of the dataset and explanation of the models used and the determination of their parameters are described. Because of the increased search space, hyperparameter optimisations were only carried out for $k = 1$. The full set of search spaces for each method and dataset are shown in Appendix D.

5.3.1 Dataset

For this chapter, strategic input 2 (consisting of sensors 1,2,3,4 heading in in the Southern direction) of intersection 115 (see Figure 5.1) is investigated. This particular intersection was chosen because it has a very low error rate across all sensors in their selected periods within the VS dataset. In the period 2015-01-01 - 2017-12-12, there are 69 error readings out of 306,621, or a 0.023% error rate.

The per phase data uses the SM dataset of the same intersection (see Figure 5.5 for the distribution of this data). It is important to note here that SCATS in Adelaide has been configured to combine readings of 0 and 1 into a reading of 0, the reason being that a reading of 1 may cause the oscillation between cycle plans (RTANSW 2004, p. 109). This data still exhibits the same varying distributions through the week and interphase random variation as seen in the VS data, (see Figure 5.6).

5.3.2 HTM and LSTM

HTM is as described in Chapter 3 and LSTM as in section 4.2. LSTM in this chapter refers to LSTM-Online from previous work as LSTM-Batch was found not to be effective when distributions in flow change. In order to determine the optimal parameters, for the two datasets, TPE is used again but with a wider hyperparameter search space as described in Table D.1. A major difference between this work and Chapter 4 is that HTM hyperparameter optimisation was carried out via TPE instead of particle swarm optimisation and searched over a greater number of the possible hyperparameters [3].

The search space for these two models is also extended to cover the inclusion of extra fields to the input of models, specifically:

[3]Source code for VS and SM hyperparameter optimisation experiments are in vs_model and sm_model folders respectively: https://github.com/JonnoFTW/htm-models-adelaide/blob/master/engine/

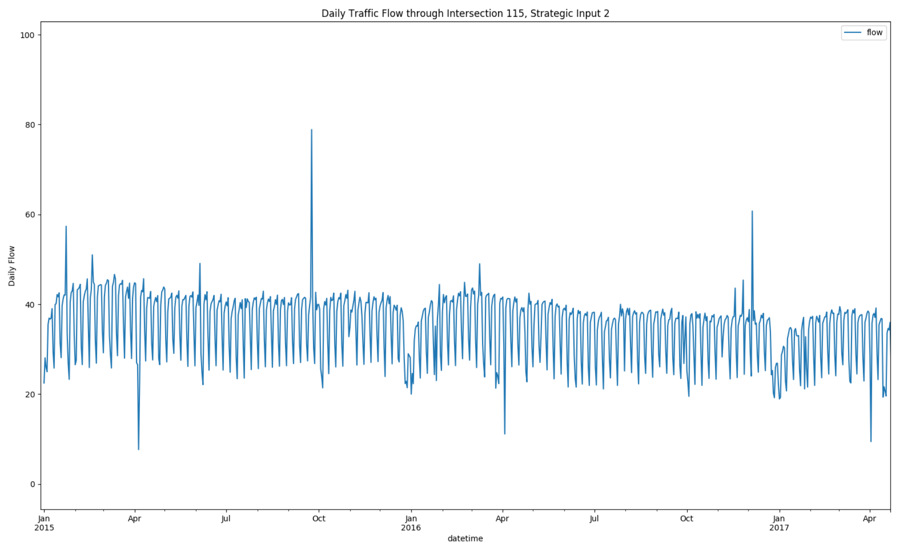


Figure 5.3: Daily Traffic Flow through Intersection 115 at Strategic Input 2

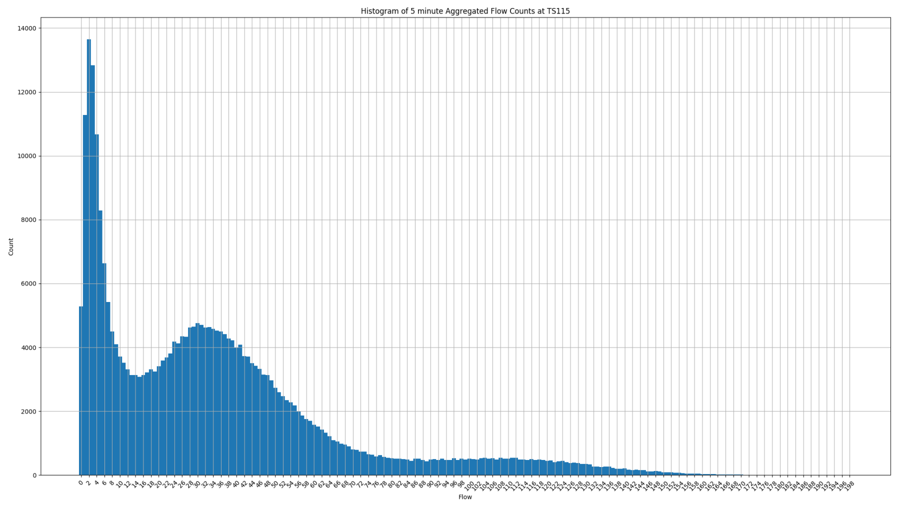


Figure 5.4: Histogram of 5 Minute Flow Counts at Intersection 115

- Weekday
- Minute of day
- Is weekend (Saturday or Sunday)
- Is a public holiday in South Australia
- Week of year
- Phase duration in seconds (for SM models)

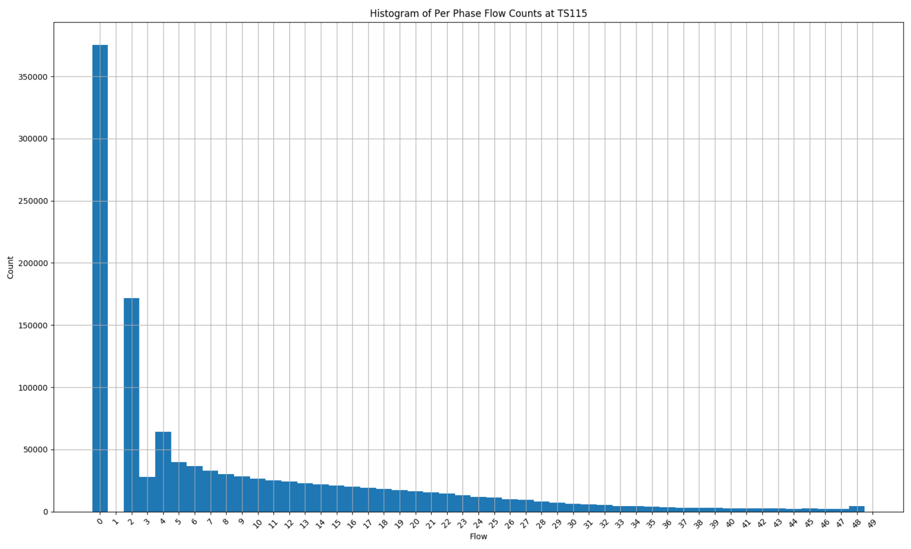


Figure 5.5: Histogram of Per Phase Flow Counts at Intersection 115

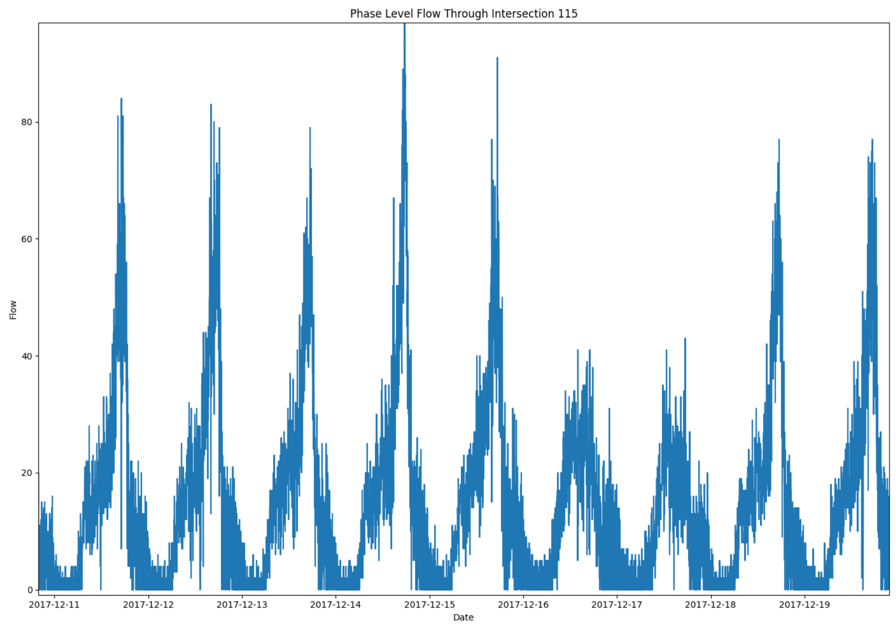


Figure 5.6: Sample of Phase Level Flow (from the SM dataset) at Intersection 115 where the change in flow between subsequent phases changes frequently in a similar fashion to how VS data changes direction frequently in 5 minute periods (see Figure 4.2a)

5.3.3 SARIMA

Seasonal AutoRegressive Integrated Moving Average (SARIMA) is a general and commonly used statistical model for forecasting timeseries data and is a combination of autoregressive, differencing and moving average models. SARIMA has the following parameters:

p the number of time lags

d degree of differencing, ie. how many previous values are subtracted from the current value in order to make the data stationary

q order of the moving average model

P,D,Q refer to the autoregressive, differencing and moving average terms for the seasonal part of the ARIMA model

m the number of samples in each season

An autoregressive (AR) model is a linear combination of past p values and is thus referred to an autoregressive model of order p (Hyndman & Athanasopoulos 2018). This can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Where ε_t is some random noise and values of c and ϕ_i are determined by standard linear regression optimisation techniques.

A Moving Average (MA) model uses past forecast errors in a linear regression to determine future values. An MA of order q can be written as:

$$\begin{aligned} y_t &= c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \\ &= c + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \end{aligned}$$

A differencing model (y_t') is the change between consecutive observations. It transforms a sequenced dataset into a stationary one. Stationary means that samples in a timeseries sequence do correlate with previous values, meaning that there are not long term patterns and with constant variance. The definition of a first order difference (where the previous value is subtracted from the current value) is:

$$y_t' = y_t - y_{t-1}$$

A differencing model of order 2 represents the difference of differences:

$$
\begin{aligned}
y''_t &= y'_t - y'_{t-1} \\
&= y_t t - y_{t-1} - y_{t-1} - y_{t-2} \\
&= y_t - 2y_{t-1} + y_{t-2}
\end{aligned}
$$

More generally, a differencing model of order d is defined as:

$$y_t'^d = -\sum_{i=0}^{'d-1} y_{t-i}$$

A seasonal model where seasons are m readings long is defined as:

$$y'_t = y_t - y_{t-m}$$

A non-seasonal ARIMA model is the combination of autoregressive (AR), differencing (I) and moving average (MA) as described above:

$$y'_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p} + \varepsilon_t$$

Where y'_t is the differenced series.

The SARIMA is written as the combination of ARIMA with an extra ARIMA component for the seasonal part (Hyndman & Athanasopoulos 2018). Typically the order of these models are selected by examining autocorrelation (a measure of the relationship between y_t and y_{t-k} for various values of k) to determine p, partial autocorrelation (a measure of the relationship between y_t and y_{t-k} for various values of k after removing the effects of lag) to determine q. In this chapter, a novel approach is taken by applying TPE to find the optimal set of orders and the selection of an additional seasonal component. The search space is listed in Table D.4.

5.3.4 Markov Model

Markov Models have a history of applications within sequence prediction for problems such as DNA sequencing classification (Salzberg et al. 1998, Henderson et al. 1997, Burge & Karlin 1998, Kulp et al. 1996) and speech recognition (Levinson et al. 1986, Rabiner 1989). Although hidden Markov models have fallen out of favour in recent years to other neural network models such as deep LSTMs (Lo Bosco & Di Gangi 2017) and deep convolutional neural networks (Rizzo et al. 2016), their previous efficacy still gives motivation to evaluate them on the short term traffic prediction task.

The Markov Model algorithm introduced here implements a n-order Markov chain (f^n) using an online transition probability updating scheme, that is, it learns the probability distribution function (P) of state transition changes in an observed sequence ($\mathbf{x}$) where a state at time t is $s_t = x_{t-n}, \ldots, x_{t-1}, x_t$:

$$f^n(s_t) \rightarrow P(x_{t+1})$$

When presented with a state (a history of traffic flow values in this case), the model will return a probability distribution over potential next state values. A new state randomly chosen according to P then becomes the prediction. If a state ($\mathbf{s}$) to be predicted on is not in the model (ie. the model has not seen that state before), then a number of fallback methods are evaluated to produce a prediction:

- Return x_t
- Return the mean of s_t
- Return the median of s_t
- Return a randomly chosen value from s_t with uniform probability

The first step involves observing a sequence and recording it in terms of a series of state transitions of order n along with the observed frequency of those transitions. For example, given an order of $n = 2$ and the sequence $x = 1, 2, 3, 1, 2, 2, 1, 2, 3$, the following state transition are observed:

$$\begin{gathered}
(1,2) \rightarrow 3 \\
(2,3) \rightarrow 1 \\
(3,1) \rightarrow 2 \\
(1,2) \rightarrow 2 \\
(2,2) \rightarrow 1 \\
(2,1) \rightarrow 2 \\
(1,2) \rightarrow 3
\end{gathered}$$

This can be represented as a transition matrix where A_{jk} is the observed frequency of transitioning from state j to state k:

$$\mathbf{A} = \begin{array}{c|ccc|}
 & 1 & 2 & 3 \\
(1,2) & 0/3 & 1/3 & 2/3 \\
(2,3) & 1/1 & 0/1 & 0/1 \\
(3,1) & 0/1 & 1/1 & 0/1 \\
(2,2) & 1/1 & 0/1 & 0/1 \\
(2,1) & 0/1 & 1/1 & 0/1
\end{array}$$

That is, when $(1, 2)$ has been observed, the probability that the next observation will be 1 is 0, 2 is $^1/_3$ and 3 is $^2/_3$.

The training and prediction algorithm used in this work is implemented in Algorithm 6[4]. The use of a map in the implementation is important especially as the number of potential states increases with the order n of the model. Given d distinct states, there are n^d possible states that would require a transition matrix of shape $n^d \times d$. For the SM data used, an order 3 model would require the storage of $3^{61} \times 61 = 13845841$ floating point values, requiring 443MB of memory to store the transition probabilities as 32bit floating point numbers. Using a map of counters, the algorithm needs only to store the $19,300$ occurring states and $107,386$ observed transitions. In the VS dataset, the flow values are also discretised to reduce the number of potential states, the level of discretisation forms one of the parameters iterated over during grid search. The search space

[4]Full code available at https://github.com/JonnoFTW/htm-models-adelaide/tree/master/engine/markov_model

for the Markov Models are listed in Table D.6.

Algorithm 6: Algorithm to approximate state transition probability distributions and make predictions on a sequence using a Markov Chain

```
input      : $sequence$ the sequence of values
parameter: $order$ the order of the model
parameter: $missingScheme$ the method used to make predictions in
           the absence of a particular state
output     : Predictions on $x_{t+1}$ for all $x_t$ in $sequence$

1  model ← empty map with default values as empty counters;
2  predictions ← empty list;
3  for $i$ ← $0$ to $sequence.length$ - $order$ do
4      state ← sequence[i:i+$order$];
5      next ← sequence[i+1];
6      if $state$ in $model$ then
7          p ← randomly choose $k$ from probability distribution,
           $k = \frac{model[state][k]}{\sum model[state].values} : k \in model[state]$;
8      else
9          if $missingScheme$ is $last$ then
10             p ← $state[-1]$;
11         else if $missingScheme$ is $mean$ then
12             p ← $\frac{1}{order} \sum state$;
13         else if $missingScheme$ is $median$ then
14             p ← median($state$);
15         else if $missingScheme$ is $random$ then
16             p ← random value in $state$
17     end
18     predictions.append(p);
19     model[state][next] += 1;
20 end
21 return $predictions$
```

5.4 Results

Models are evaluated here using the RMSE score as described in section 4.3, where 0 indicates a model whose predictions are exactly those observed in the next timestep. Increasing RMSE values indicate models who comparatively make more inaccurate predictions. For the two tasks involved: next phase volume prediction (from SM data) and predicting aggregated total flow in the next 5 minutes, (from VS data) ideal predictors will:

- Continuously learn overall distributions and adjust when they change,
- Learn and predict accurately the seasonal and rare variations in distribution such as public holidays,
- Predict the frequent oscillations in volume direction between individual readings (see Figure 5.6).

RMSE is selected here as the evaluation metric (over those described in section 4.3) for 2 reasons:

- Most other literature publish model scores as RMSE
- RMSE does not suffer from a relatively high error score at low flow values. As there are times of zero vehicle flow, percentage based scores such as MAPE give a near infinite error value when the real value is 0 (or near-zero when using a small epsilon value to ensure this doesn't happen). When $y = 1, Y = 0$, $MAPE = \inf, RMSE = 1$, thus a prediction of 1 during zero flow according to RMSE is very good, but infinitely poor according to MAPE.

It must be noted that when comparing models here with models in other work that total vehicle flow and standard deviations must be taken into account, as these are unique to each intersection. That is, the best score attainable by a particular model at a particular intersection in one city, may be different for the same model at another intersection in a different city.

The best error score for each algorithm on each dataset are shown in Table 5.1. Tables of hyperparameters and grid search spaces are shown in Appendix D.

A n-shift model performed the worst of all models even with grid search over $n \in \{1, \dots, 15\}$ as would be expected of such a simple model. The more complex models performed better as they have mechanisms to automatically learn data distributions.

The HTM models (Figure 5.7, Figure 5.13) both effectively learned to predict the highly variable data and adjust when encountering holidays, as seen in

Algorithm	**Best RMSE**	
	VS	SM
HTM	9.32	5.79
LSTM	9.10	6.27
SARIMA	9.43	5.16
x_{t-n}	11.01 $(n = 2)$	7.234 $(n = 3)$
Markov	**6.99**	**4.887**
Rolling Mean	9.84	5.495

Table 5.1: Error scores for best models (as determined by hyperparameter search) for each algorithm and dataset

Chapter 4. The predictions made by HTM were comparable to the both LSTM and SARIMA in terms of error score.

The LSTM models (Figure 5.8 and Figure 5.12) both eventually learned to approximate a rolling average. The optimal VS LSTM model (see Table D.2) was a 3 layer network with decreasing LSTM layer sizes (176, 88 and 60 cells each), indicating that there is a higher order pattern to the data (ie. some degree of seasonality). The optimal SM LSTM model (see Table D.3) was a single layer of 212 units and more closely resembled a rolling mean than the VS model, suggesting that future work may involve forcing the SM model to have 3 or more layers.

A close inspection of the SARIMA model's prediction in Figure 5.9, reveals that it does indeed closely follow a rolling mean over the last 8 elements Figure 5.15. This behaviour, while achieving a competitive error score, is not desirable.

Markov Model with mean fallback had the best prediction score in both tasks (see Figure 5.10. Figure 5.16). The best model for SM data an order of 6 and a mean fallback scheme (see Table D.6). Two observations are made when closely examining the output of this model (Figure 5.11):

- The Markov Model predicts effectively at low flow periods, resulting in a low error score during these periods.
- The fallback method (employed mostly at high volume times, when changes in flow are erratic) as the mean of the previous 6 values is somewhat accurate, although this behaviour is not desirable.

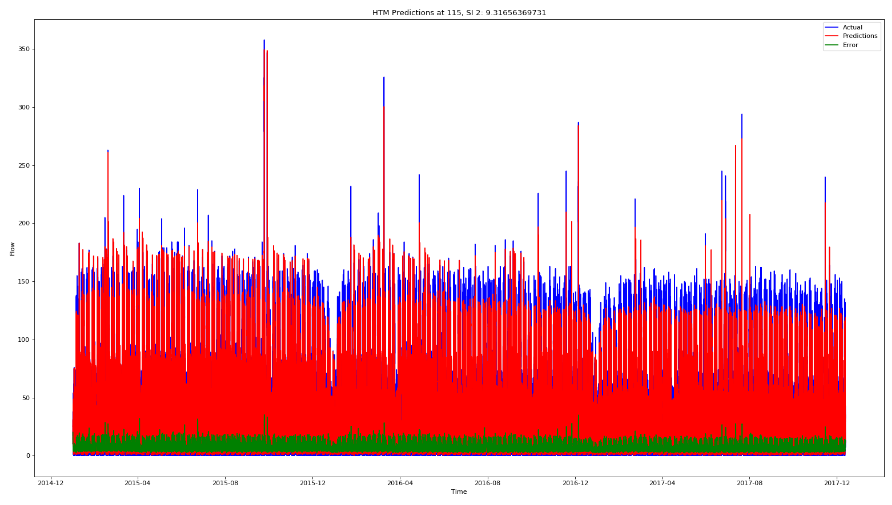


Figure 5.7: Predictions from the best HTM model on VS data. Shows error scores over time, which peak during high volume periods

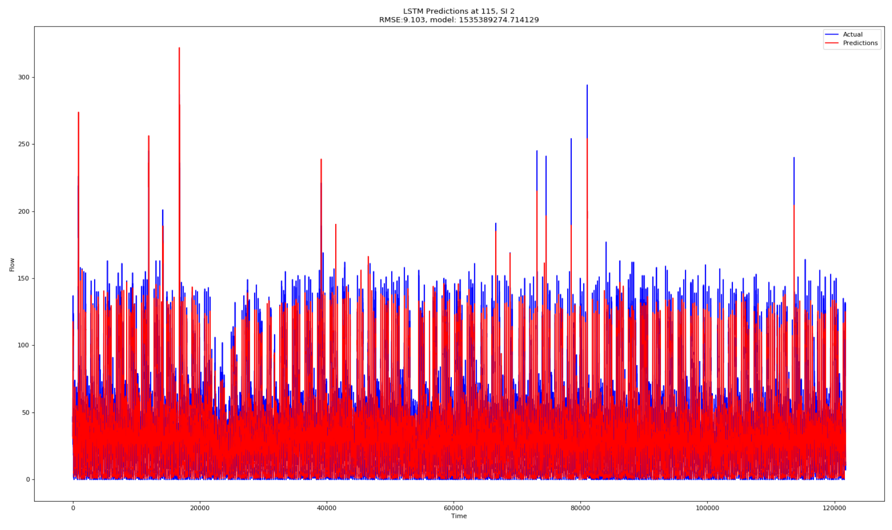


Figure 5.8: Predictions from the best LSTM model on VS data

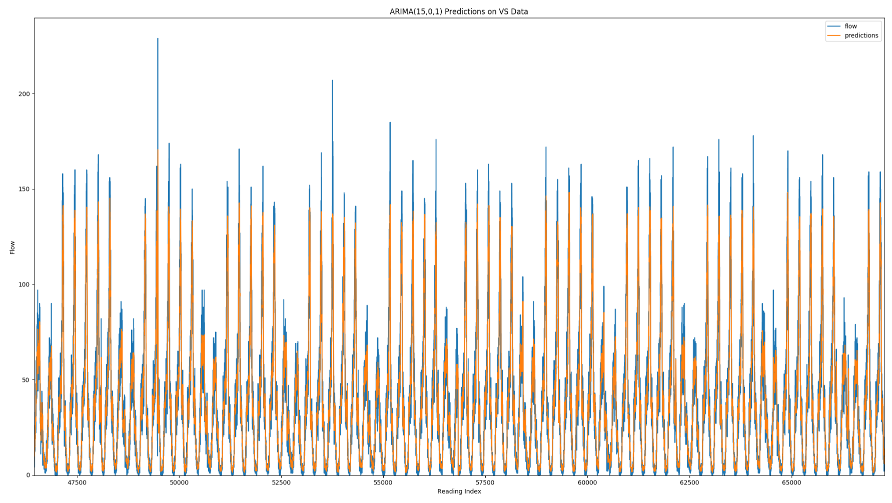


Figure 5.9: Predictions from the best ARIMA model on VS data

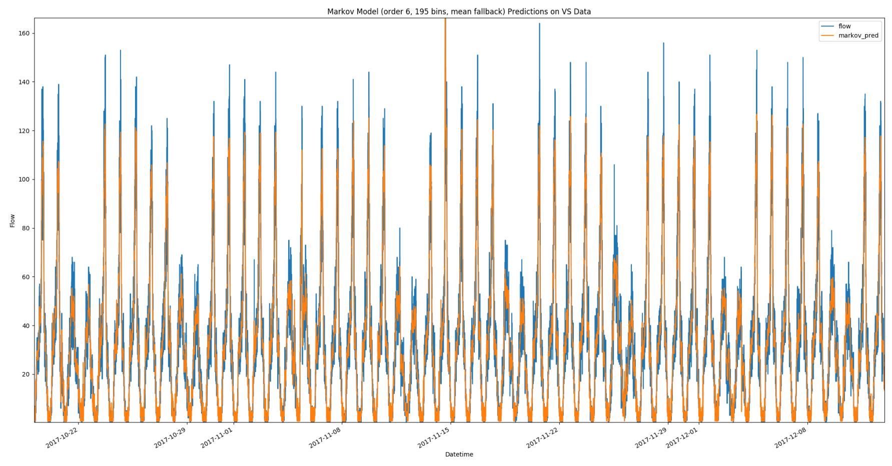


Figure 5.10: Predictions from the best Markov Model on VS data

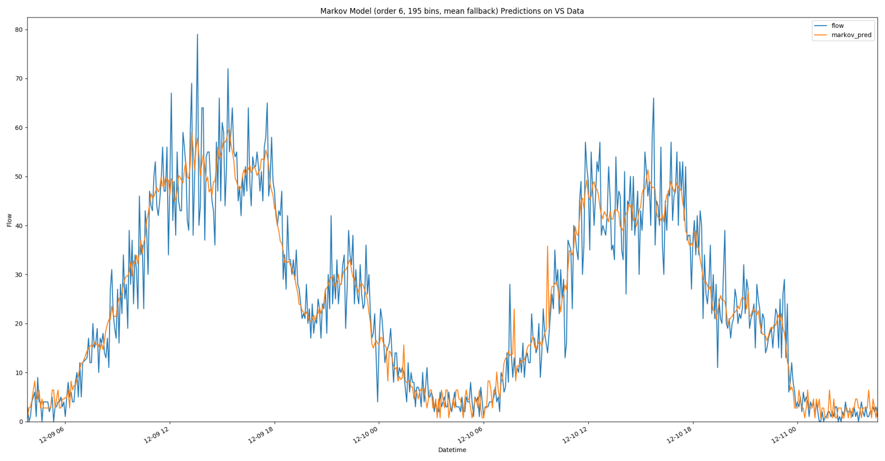


Figure 5.11: Closeup of Markov Model Predictions on VS Data

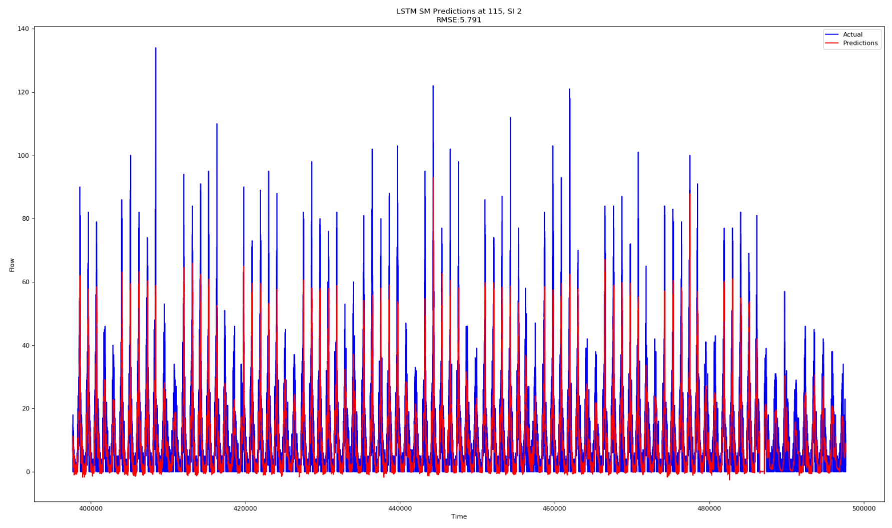


Figure 5.12: Predictions from the best LSTM model on SM data

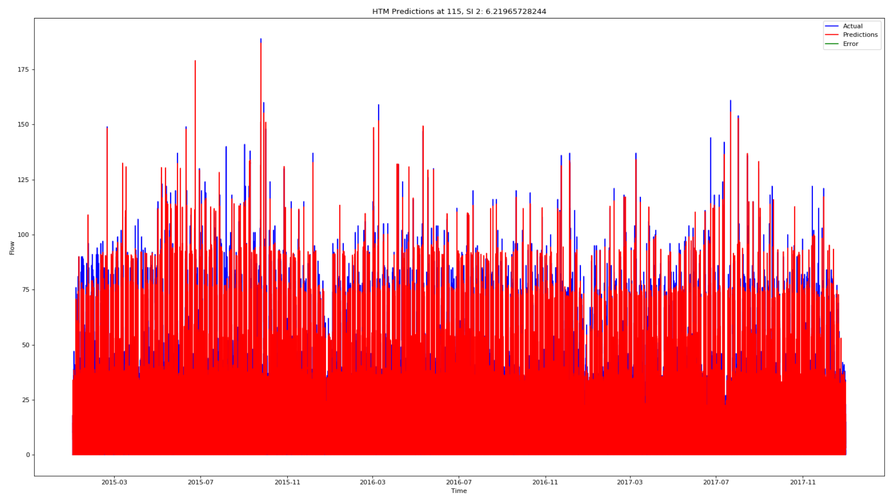


Figure 5.13: Predictions from the best HTM model on SM data

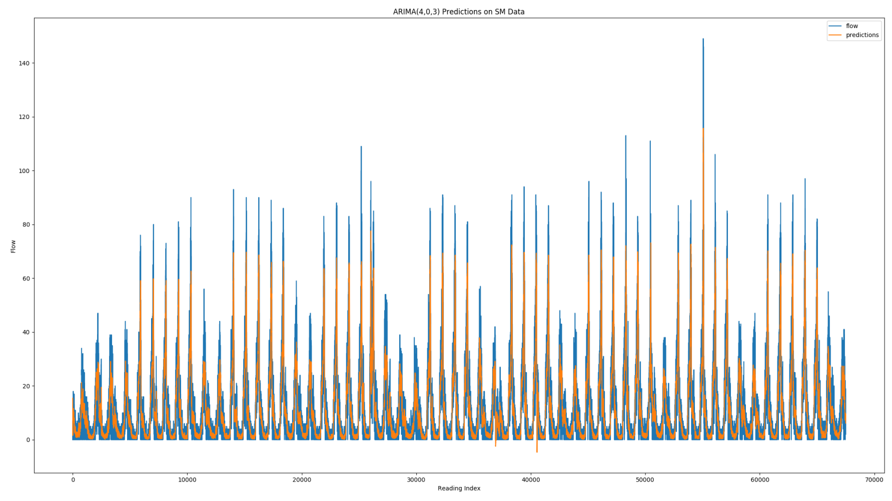


Figure 5.14: Predictions from the best ARIMA model on SM data

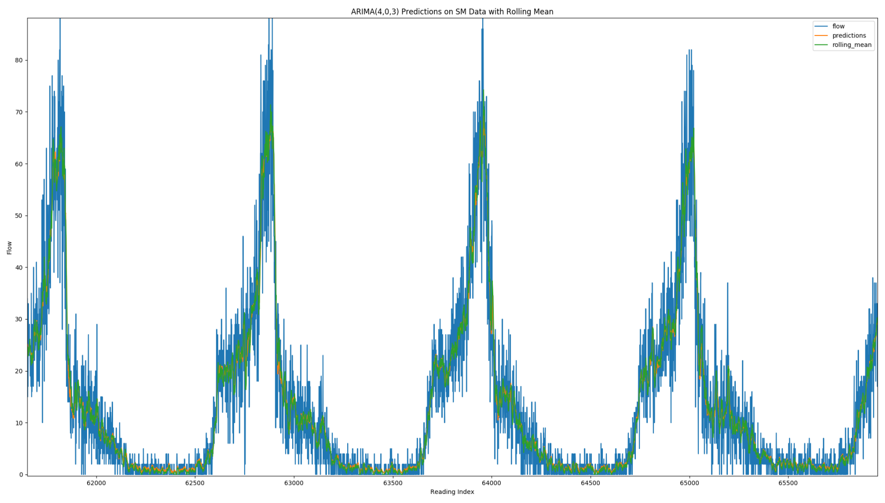


Figure 5.15: Comparison of Rolling Mean ($n = 8$) and ARIMA(4,0,3) on SM Data

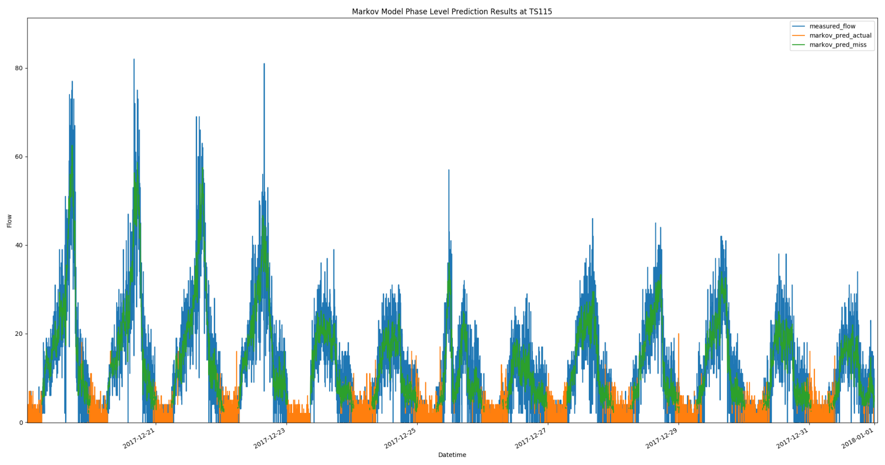


Figure 5.16: Markov Model Predictions from the best Markov model on SM data. Orange data points show when it made a prediction from its observed state transition probability distributions, green data points show when it used the mean fallback mechanism

5.5 Conclusion

This chapter gave an overview of some hyperparameter optimisation techniques and their previous use in optimising models for short term arterial traffic prediction. Such techniques have been effective at significantly improving model performance and detecting novel model structures. Most previous research focuses on aggregate flow prediction (in limited spatiotemporal contexts), and so this research which expands into predicting traffic volumes through a particular movement in the next phase is novel in the current literature. A novel Markov Chain model is presented with superior results compared to the other models tested, although it did not exhibit desired behaviours of emulating the frequently changing flow between readings.

TPE hyperparameter optimisation was shown to be effective at improving model performance of HTM (where previous work optimising a small subset of HTM parameters using particle swarm optimisation), LSTM and SARIMA. Furthermore, it could effectively select the optimal order for SARIMA models, without human input, automatically determining that adding a seasonal component to both models would not improve performance (see Table D.4). A significant drawback is both the LSTM and ARIMA models tended to approximate the rolling mean (see Figure 5.15), and despite their competitive RMSE score, did not effectively model the frequent oscillations of the real data, which would not be useful when automatically adjusting cycle plans.

HTM and LSTM models did effectively model the frequent changes in data and overall trend, their predictions were still not entirely accurate, although HTM is the preferred model as the others all had issues of estimating a rolling mean. The proposed Markov Model with fallback achieved superior performance to all other models, due to its ability to switch between a learned sequence prediction, and rolling mean. With appropriate extensions, the Markov Model could overcome issues where it does not accommodate changes in distribution over long periods.

5.6 Future Work

Future work in the area of meta-optimised short term traffic prediction models could investigate:

- Increasing the search space for multivariate models over surrounding sensors, beyond those of upstream and downstream intersections. It may be the case that for some intersections, their future flow may be correlated with the flows of nearby intersections. Identifying such relationships (should they improve model performance) may also be of great interest to traffic engineers when planning road improvements or devising intersection cycle plans.

- The direction change in data may be of interest in feature search spaces, ie. a field which indicates the sign of the change between x_t and x_{t-1}: $\mathrm{sgn}(x_t - x_{t-1})$ with values in $\{-1, 0, 1\}$ or a history of such sign changes. For a Markov Model, this means that the set of potential next flow states are filtered by those matching the direction of a randomly selected sign, where the sign is randomly chosen with probability according to the observed sign change frequency.

- Extending the Markov Model to accommodate changes in overall distribution. Potential methods by which this can be achieved include: adding a small decay factor over all transition probabilities, maintaining a history of observed state transitions and decaying those that are not used or recalculating transition probabilities from a temporal history window (although this would come at a significant memory and computation cost).

- Using evolutionary methods to build up a neural network model (that may or may not include various recurrent layers such as LSTM or Convolutional LSTM). The efficacy of such methods were shown by Real et al. (2017) for a 10 class image classification task and may be transferable to time-series regression tasks such as short term arterial traffic prediction.

- Evaluating these models again on phase level data that contains flow counts of 1 to determine if such volumes can improve model performance (although this would require adjustment to the ITS data collection software). Further investigation should also investigate models that predict at increasing values of k, with multiple outputs where models permit, across a greater number of intersections and strategic inputs.

Chapter 6

Arterial Incident Detection via Anomalies

Given the datasets introduced in section 2.5, the tasks approached in this chapter differ greatly from those traffic flow volume tasks tackled and contributed to in Chapter 4 and Chapter 5. In this chapter, the tasks moves away from the problem of flow prediction and into utilising observed flow values to determine the occurrence of incidents within arterial road networks. Whereas much of the previous work approach this task in a freeway context, the work presented here contributes towards incident detection in arterial traffic, evaluated over long time periods while developing tools that could facilitate different approaches in the future.

This chapter discusses:

- The aims of arterial incident detection
- A review of techniques for detecting outliers in stream data
- A review of techniques for road based incident detection
- An investigation into the novel application of anomalies to incident detection and an evaluation of this approach

6.1 Background

Roadway incidents refer to those non-recurring events that result in increased traffic congestion (Parkany & Hall 2005) by partially or completely blocking the flow of traffic. Examples of incidents are:

- Accidents, breakdowns and illegal parking.
- Natural events such as flooding and landslides.

- Spilled loads and debris.
- Public events such as protests, riots or fairs.
- Unscheduled infrastructure maintenance.
- Signal malfunctions or improper signal configuration.

During an incident the capacity of the road is restricted, resulting in queues and delays which have a wide variety of consequences including congestion which can result in traffic jams, increased pollution, increased travel cost, fuel wastage, reduced productivity and a danger of secondary incidents. Prompt and accurate incident detection is vital in reducing congestion and improving emergency response times (Parkany & Hall 2005). According to Farradyne (2000), incident based congestion is responsible for 50%-60% of all delays. This should not be confused with recurrent congestion, which is known, normal, and does not warrant an emergency response. A prime example of this peak-hour traffic exceeding roadway capacity (Zhang 2005). This issue is a significant motivator in developing systems that can prevent incidents from occurring, and rapidly detecting and responding to them when they do.

The task of detecting these non-recurrent congestion events (NRCs) is difficult because of the heterogeneous nature of the urban road network and its utilisation (Anbaroglu et al. 2014):

- Links are of varying length, speed limits, and typical commuter lane usage.
- Network usage across time varies, ie. weekday morning traffic is different from weekday afternoon, which is entirely different to weekend and public holiday usage. Thus the distribution of incidents throughout time will vary along with their implications for nearby traffic flows.
- Evaluation of detection methods is difficult because of the lack of ground-truth data. That is, it is difficult to confirm whether or not an NRC is occurring in reality, and so prediction evaluation is difficult. This can be mitigated (Cheu & Ritchie 1995) by using accident datasets, although not all NRCs are caused by accident (and recorded as such) per se. Additionally these datasets may not be transferable between locations. Since the ramifications for traffic flow from an accident in one time and place cannot be used to infer the occurrence of an incident in another due to the unique nature of the road and its usage at the time.

Given the above difficulties, the following observations can be made that can help drive the development of systems to detect NRCs:

- Ideally individual models would be trained for specific locations to ensure that individual distributions in traffic flow are captured and minimise the need for human intervention when developing models considering that cities often have hundreds of signalised intersections, each with their own unique usage and NRC impacts.

- Comprehensive datasets covering the occurrence of all types of NRCs (or as is practically possible to collect) must be created in order to aid the development of models that learn from previous data.

6.2 Automated Incident Detection for Arterial Road Networks

In this section only those Automated Incident Detection (AID) algorithms that are applicable to arterial road networks are examined. This is because the vast majority of AID methods are made for freeway traffic and are not transferable to the more complex arterial network. Moreover, the discussion is narrowed to those algorithms that can be applied to loop-detector data, as some algorithms use other forms of vehicle detection such as image processing, probe vehicles and emergency call monitoring (Parkany & Hall 2005).

Previous reviews into state-of-the-art AID algorithms are listed in Subramaniam (1991), Stephanedes et al. (1992), Balke (1993),Mahmassani et al. (1998), Peterman (1999), Black & Sreedevi (2001), Black & Sreedevi (2001) and Parkany & Hall (2005). These reviews also cover algorithms outside of loop-detector methods.

Abdulhai et al. (1999) covers the development of algorithms that use adaptive algorithms, specifically those that use artificial neural networks, and briefly evaluates their effectiveness and how their implementation can be improved using genetic algorithms.

The investigation of methods for AID in arterial road networks started in the late 1980's (Bell & Thancanamootoo 1988). This study used data collected by SCOOT in London and Middlesbrough which continuously monitors loop detector data. Additionally, MULTISIM was used for microscopic traffic simulation.

Their algorithm uses a continuously updated, exponentially smoothed average and variance of the cyclic occupancy (the percentage of time a vehicle is over a detector per cycle (Ivan et al. 1995), which correlates negatively with speed and positively with flow) which is derived from flow, vehicle length, loop length, velocity, velocity variance and covariance of velocity and vehicle length. Upper and lower bounds are established as the exponentially smoothed mean and variance, anything outside these limits is an incident.

The results of simulation are not well described, they use no metrics that are developed in later papers so are not readily comparable to other methods. Two intersections were tested and could detect an incident initially, but the algorithm eventually classified the disruption as normal and no longer indicated an incident. The authors admit that this algorithm is suitable when it can be configured on a per intersection basis and only at those intersections deemed critical.

Han & May (1989) uses an expert system methods, where alarms are defined by human experts as a series of "if-then" statements, a task which can be tedious and error prone. They system developed (called `TOPDOG 1.0`), was evaluated on real-world flows on weekdays, from 5pm until 9pm but no actual metrics are provided by Han et al. Modern predictive systems have mostly abandoned expert system methods (in favour of neural network based models) due to their inability to learn autonomously or automatically adapt to changes in input (or simply that the system did not encapsulate some cases) (Leith 2010).

Ivan et al. (1995) use fused data from both loop detectors and probe vehicles. Separate algorithms (these are not described by the authors) process each of the two data sources separately, which are then fused together by a feedforward neural network trained by back propagation. Training data is generated using 100 different micro simulations with the ADVANCE simulator on a 5km section of major arterial streets with 39 loop detectors at 11 intersections and one in four cars as probe vehicles. The two neural networks tested were:

- Algorithm Output Fusion (AOF): in which the outputs of the fixed detector algorithm (FDA) and probe vehicle algorithm (PVA) are fed into the network to produce a combined output that indicates incident probability.
- Integrated Fusion (IF): in which loop detector and probe vehicle data is fed directly into the network to produce an incident probability output.

Algorithm: Dataset	Detection Rate	False Alarm Rate
AOF: Training	81.50%	0.11%
AOF: Reserved	100.00%	0.00%
IF: Training	65.00%	0.54%
IF: Reserved	70.00%	0.96%
Fixed Detector Algorithm	65.90%	0.00%
Probe Vehicle Algorithm	53.70%	0.00%

Table 6.1: Results of Ivan et al. (1995)

The results summarised in table 6.1 show that the AOF outperforms IF, FDA and PVA. IF had a poor detection rate of 70% after training with an unacceptable 0.96% error rate. AOF achieved a DR of 100% on the reserved data with an error rate of 0.00% while the training data had 81.5% DR, the authors credit this discrepancy to the types of crashes used in the partitioning being too similar, ie. the reserved dataset used more extreme examples. This could have been mitigated by the authors if they averaged results over several randomised test/train partitions.

This work is expanded on by Ivan (1997), in which several neural network variants are tested using the described data fusion framework:

- Two input network, the network used in the previous paper.
- Output memory network, utilises a memory unit that returns the sum of every previous network output value weighted by one-half for every time interval since the value was computed. The memory unit is an additional input.
- Adjacent link network, which takes 6 inputs, 3 scores from both FDA and PVA from upstream, downstream and target links.
- Full network: combines the adjacent link network with an output memory module.

Training data was generated by simulating traffic flow in suburban Chicago using the INSTRAS simulation program with the same properties as the previous paper. As in the previous work, the author's random partitioning of the data into training/train sets resulted in more severe incidents being placed into the testing, which resulted in experimental results showing a higher DR on the test data partition. The full network outperformed all other tested networks with a 92.5% DR on training data and 100% DR on test data, and 0% FAR for both partitions. Although there is improved performance, conclusions about the transferability of these algorithms is not reliable due to the way the experiment was performed. Additionally, in the absence of real world data, the authors may have introduced significant bias in their simulation-based dataset, causing the neural network to only be effective at detecting certain types of incidents.

Stephanedes & Hourdakis (1996) investigate the transferability of three freeway incident detection algorithms to the arterial road space. They test on two sites I-35W in Minnesota and I-880 in California using the **DE**tection **LO**gic with **S**moothing (DELOS) (Chassiakos & Stephanedes 1993, Stephanedes & Chassiakos n.d.) algorithm. DELOS works by smoothing (using a low pass filter (Stephanedes & Chassiakos n.d.)) the raw sensor data to remove random fluctuations, pulses and compression waves. An incident is detected when an adjacent detector station finds a significant difference in the filtered occupancy of the link in a short period of time. The method was tested in comparison to two other comparative type algorithms (CALIFORNIA and Algorithm#7 (Payne & Tignor 1978), a decision tree system) on a dataset of 159 incidents over 20km. DELOS achieved a DR of 89% compared to California's 72.4% and Algorithm#7's 76.33%.

The drawback of these algorithms is that they require considerable configuration for every detection site.

Weil et al. (1998) use neural networks and fuzzy logic for incident detection. They evaluate their use against McMaster (Persaud et al. 1990) incident detection algorithm as a baseline. A drawback of the McMaster algorithm is that it relies on temporal data only, and not spatial differential data, and so has difficulty differentiating between changes in traffic due to factors such as weather conditions which are distinct from those caused by incidents.

Khan & Ritchie (1998) investigate the use of multilayer feed-forward neural networks. They examine three types of problems: lane blocking incidents, special event conditions and detector malfunctions. They describe the problem of AID as a pattern recognition and classification one. A stochastic microsimulation was run using the NETSIM traffic simulation system and the interlane interactions were observed as a result of a variety of different incidents. The output of the described ANN was a simple binary classification: 0 for non incident, and 1 for incident.

The simulated incidents blocked between 1 and 3 lanes and went from 2 to 16 minutes in length. The ANN achieved a poor detection rate of approximately 67% for 1 lane at all time lengths and 100% detection for 3 lane blocking with 0.23% FAR and 1.16% FAR respectively.

Thomas (1998) addresses the problem of AID as a *multiple attribute decision making problem with Bayesian scores*. Her system uses data collected from link wide detectors (to gather link travel times) and station detectors (to measure intersection occupancy). These 2 data vectors are then used to calculate a score for the link and station, the smallest of which is then fed to a Bayesian classifier to determine the traffic state (incident or non-incident).

The data used in her experiments is derived from a microsimulation of 9 multi-lane intersections using the INTRAS simulation software. The system achieved a FAR of 1.45% and a classification rate of 96%.

Yuan & Cheu (2003) use several Support Vector Machines (SVM) with non-linear kernel functions to perform AID on arterial and freeway traffic. They simulated 648 different incidents across 9 links using the INTEGRATION traffic simulator and compared their accuracy to that of a multilayer feed forward neural network and probabilistic neural network as baselines.

The input vectors consisted of upstream and downstream volume of each lane for the current and 3 previous cycles. Two different SVMs were used in testing, one with a polynomial kernel function (SVM_P) and one with a radial basis kernel function (SVM_RB), all implemented in the SVM MATLAB toolbox. These models were compared with Probabilistic Neural Network (PNN) and Multilayer Feedforward Network (MLF) on the same training data. The SVMs outperformed the neural networks, getting drastically improved detection rates on single lane blockages, 80.2% for SVM_P, 74.7% for SVM_RB, 27% for MLF and 61% for PNN. The neural networks performed better on multi-lane blockages although still below the SVMs, 97.5% for SVM_P and SVM_RB, and 93.2% for MLF and PNN. These findings suggest that pattern recognising machine learning methods (specifically SVMs) can be useful for performing arterial AID on both single and multi-lane blockages, although it is not apparent whether they can be applied to real world datasets.

Zhang (2005) and Zhang & Taylor (2006), attempt to make freeway algorithms transferable to arterial road networks with the TSC_fr algorithm. This solution employs Bayesian networks, which attempts to replicate the rule based reasoning used by experts in a traffic management centre. Essentially, they reason that:

- Low upstream volume and high upstream occupancy indicate either bottleneck or back propagated congestion.
- If the downstream volume is low and downstream occupancy is high as well, there is back propagated congestion.

What constitutes high, medium and low occupancy and volume for a particular time and place is learned from historical data and used to make a decision about new data by calculating a probability of a reading indicating an incident. Additionally, Zhang (2005) introduces a framework for incident detection, with a focus on algorithm transferability.

Zhang et al. (2007) reportedly improve on their previous work by adding some additional optimisations to the Bayesian network used, although this updated model is then tested in a micro simulation of 3 intersections and 3 incidents which achieves a 100% detection rate. The new system had the same DR (100%), improved FAR (2.22% $\rightarrow$ 1.85%) and MTTD (127s $\rightarrow$ 80s) over the old system.

Viswanathan et al. (2006) use fuzzy association rule mining using the adaptive-network-based fuzzy inference system (ANFIS) to fuse numerous data sources. The algorithm generates a number of IF-THEN rules, which are then used to classify traffic flow from probe vehicles. This method was tested using inputs from a 30 day period and achieved a DR of 92.72% and FAR of 11.54%. Ivan et al. (1995) cites this failure rate as unacceptably high, although the original author thinks it is acceptable.

Chen et al. (2007) use neural network ensembles to perform AID. This technique involves training up numerous neural networks and then combining their results which provides additional performance over a single neural network. The individual models within the ensemble are MLP networks, based on the work of Yuan & Cheu (2003). The analysis used freeway data from I-880 freeway in San Francisco with 45 incidents in 1993 morning and evening peak periods. The networks were trained on 6442 instances of which 2100 are incidents, with each instance covering 7 features including: speed, volume and occupancy upstream and downstream of the detector station with a class field for incident or non-incident.

The testing method used consisted of 5 different MLP networks being used individually and then in an ensemble. The ensemble was trained using bagging and outputs combined using boosting. The ensemble had a DR of 86.96%, FAR of 3.04% and MTTD of 3.1 minutes; these scores were an improvement on the average performance of the individual networks, although the average MTTD for the individual networks was lower at 2.63 minutes. Though some networks scored higher and some lower than the ensemble.

Although this was only tested on freeway traffic, it appears robust enough that the techniques could be expanded to work on arterial traffic, based on the fact that previous research has investigated the use of neural networks (Yan & Han 2003, Khan & Ritchie 1998, Ivan et al. 1995, Ivan 1997) for arterial AID, though none have utilised boosting and bagging yet.

Šingliar & Hauskrecht (2010) proposed to detect incidents from noisily labelled data using a learner based on SVM. Due to the noise inherent in the large volume of training data of traffic sensor measurements, they use incident label realignment (as often the recorded time of an incident did not properly align with when it actually occurred) using a dynamic Bayesian network to estimate the incident time which leads to improved incident classification performance. The motivation for their research is to move away from AID methods that require extensive tweaking of location specific parameters and thresholds, and expert knowledge. Additionally they sought to improve on previous supervised techniques that, while showing good results, suffered from overfitting due to the small training dataset.

Before label realignment was performed, the SVMs were compared to dynamic naïve bayes (DNB) using the Activity Monitor Operating Characteristic (AMOC) curve metric to simplify results as this is more suitable to evaluating the detection of rare events. The SVM scores 0.693 AUC while DNB scored 1.094, where a smaller AUC is better.

Lee et al. (2011) uses spatiotemporal bottleneck mining. They source their data entirely from location-based service (LBS) based applications, so that any vehicle using the service becomes a probe vehicle. The system combines data from individual vehicle journeys, traffic network snapshots and uplink information.

The prototype system was tested on a fleet of 500 taxis within the urban network of Taipei City and was operational from February 2006 to March 2007 with 0.5 million uplink reports per day in 30s intervals. Only peak hours in AM (0730-0930) and PM (1730-1930) as these periods have reasonably high congestion. Numerous heuristics are compared to detect bottlenecks, the best was Congestion Propagation Heuristic (CPH) showing 79% accuracy on weekdays and 72.1% accuracy on weekends with a standard deviation of 0.055 which was a significant improvement over the others tested. CPH uses the idea that traffic demand of a congested area propagates to nearby congested areas) was less accurate on weekends due to the instability of data during these periods, ie. one weekend's traffic patterns may differ significantly to those of another.

Anbaroglu et al. (2014) describes the use of spatiotemporal clustering to perform AID. They use four separate pieces of data to identify NRCs:

1. Link Journey Times (LJTs)(Robinson & Polak 2006) collected by Automatic Number Plate Recognition (ANPR); this allows operators to estimate travel time between two locations.
2. An adjacency matrix of the road network
3. Historic LJT data used in statistical analysis to estimate the present LJT.
4. Congestion factor used as a factor to multiply by the expected LJT to identify excessive LJT. This input requires expert knowledge though.

The output is then clustered and a metric of severity and duration derived from the episode of NRC where severity indicates the total excess LJTs. The spatio-temporal element can allow the detection of NRCs that span multiple links as they will have similar flow disruption as their LJTs will overlap. When this overlap occurs, it can be clustered and an incident can be inferred in both time and space.

The main drawback of this spatio-temporal method is the requirement of expert input for the congestion factor. The detection of episodes is highly dependent on the given congestion factor.

6.3 Stream Mining

6.3.1 Stream Mining for Outliers

In stream mining, an outlier is defined as those events that are rare, occurring with frequency that is dependent on the application domain, though typically they are in less than 5% of all readings (Pokrajac et al. 2007). These readings should not be confused with errors (readings that indicate a sensor has not collected valid data) and noise (readings from a faulty sensor that has not disclosed itself as being in error)(Zhang et al. 2010).

Pokrajac et al. (2007) uses incremental Local Outlier Factor (LOF) (Breunig et al. 2000) for a density based stream outlier detection algorithm. LOF is a not a binary classification (ie. outlier versus non-outlier), but rather a degree to which a point is considered a local outlier. In the proposed incremental LOF algorithm, LOF is calculated as a ratio of the average local reachability density of its k nearest neighbours and local reachability density of point q where:

$$lrd(q) = \frac{1}{\sum\limits_{p \in k\mathrm{NN}(q)} reach - dist_k(q,p)/k}$$

$$\mathrm{LOF}(q) = \frac{\frac{1}{k}\sum_{p \in k\mathrm{NN}(q)} lrd(p)}{lrd(q)}$$

As new data points are inserted, old data points are deleted, which improves performance. Data points themselves are labelled with LOF that indicates their degree of being an outlier. Points with smaller neighbours have a higher factor than those in large neighbourhoods, so outliers are those points with a LOF below a certain threshold. The advantage of this algorithm is that outliers are found with respect to their neighbours and not the global model, and allows outliers to be found in a dataset with multiple densities. This final property is not pertinent to the traffic dataset as its density is uniform because it arrives in 5 minute intervals and the slowing of traffic after an incident is quite sudden.

Jain et al. (2006) propose a non-linear stream clustering algorithm that adapts to changes in the evolving high-dimensional stream. They analyse first separate the data stream into partitions using a novel kernel method approach. These partitions are then mapped into a low-dimensional space before assigning them to a cluster. Their method is successfully applied to a number of real-world datasets, including news documents and network intrusions. They report that within their algorithm, any cluster that contains a single point only is an outlier, which could potentially be of use, although no detail is provided about its effectiveness in detecting such outliers.

In order to address the typically high computational cost and lack of explanation in density based outlier detection, Abe et al. (2006) uses classification of a labelled dataset containing artificially generated examples to simulate outliers to generate a training set. A selective sampling technique based on active learning is then employed to perform the classification.

Silva et al. (2013) surveys stream data mining algorithms and identifies the following clustering algorithms that can detect outliers in large data streams:

BIRCH: first introduced by Zhang et al. (1996), Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), aims to find clusters in large datasets in a single pass. Provided a dataset of multi-dimensional points and target number of clusters k, BIRCH has 2 main phases:

1. Create a Cluster Feature (CF) tree, where a cluster has N d-dimensional points ($\bar{X}$) and a CF is a triple consisting of: number of points in the cluster, linear sum of points (encapsulating the location of the cluster), and square sum of data points (encapsulating the spread of the cluster), ie. CF $= \left(N, \sum_{i=1}^{N} \bar{x}_i, \sum_{i=1}^{N} \bar{x}_i{}^2\right)$. The CF tree is a height

balanced tree with nodes containing a CF and pointers to at most B (where B is the branching factor) child nodes. New data points are added to the tree by descending the hierarchy and selecting child nodes based on closest linear sum. Once a leaf node is reached, the point is added to the CF and a new child created if its radius now exceed a threshold T.

2. Global Clustering: where an existing clustering algorithm (such as k-means) is used to cluster subclusters at the leaf nodes of the CF tree. Points that are too far from their resultant cluster are considered outliers.

This method is highly efficient as it only stores clusters as a summary of data into which new points can constantly be added and the summary updated. The downside here is that in order to identify outliers in incoming data, the global clustering algorithm must be run for every new data point.

CluStream: Aggarwal et al. (2003) introduce CluStream as a framework for clustering data streams with a two step process: online micro-clustering and online macro-clustering. Their online component uses CFs for both the datapoints and their corresponding timestamps to summarise a set of observed datapoints. As new datapoints are added, they are merged into existing cluster or a new is created in a similar manner to BIRCH. The offline component uses a weighted k-means algorithm on the microclusters to obtain the clusters for the current timestamp. Microclusters are removed after after a specified period has passed. Outliers are obtained in the same manner as BIRCH does.

DenStream: Cao et al. (2006) expands on the previous methods by allowing the creation of an arbitrary number of dense micro-clusters with arbitrary shape (previous methods only allow for a fixed number of spherical clusters). This algorithm uses a density based clustering algorithm to group multiple core-micro-clusters (using CF tuples) with a fading function for their associated timestamp $f(t) = 2^{-\lambda t}$ into micro-clusters. Micro clusters that do not meet a weight threshold in their CF are designated outliers.

D-Stream: Chen & Tu (2007) uses a density based approach in which incoming points are placed onto a grid and the density of points within all grid cells used to determine the presence of a cluster, outliers are those cells with insufficient density. The recorded density of each grid cell is decayed over time to reduce the importance of historical datapoints, meaning that many resultant clusters can appear and fade over time.

6.3.1.1 Anomaly Detection In Streams

Ahmad & Purdy (2016) provides a survey of the the state of the art methods for anomaly detection in streams. They are:

- HTM (Cui et al. 2015) as described in Chapter 3
- Seasonal Hybrid ESD and Seasonal ESD introduced by Hochenbaum et al. (2017), described in subsubsection 6.3.1.2
- Etsy Skyline by Etsy (2015). Skyline is an ensemble method that calculates a score based off an average of binary scores from multiple simple methods: median absolute deviation, Grubbs' test, standard deviation from average, standard deviations from moving average, mean subtraction cumulation, least squares prediction and histogram bins. [1]
- Bayesian online change point detection introduced by Adams & MacKay (2007)

6.3.1.2 Seasonal Hybrid ESD

Hochenbaum et al. (2017) presents two methods for anomaly detection in streams: *Seasonal ESD* and *Seasonal Hybrid ESD*. Motivated by the increasing need for uptime and reliability in social network applications, where user demand is seasonal, the novel methods introduced are able to detect anomalies when the data has a seasonal component. Critically, they show that SHESD is effective at detecting anomalies in real world production data.

Given a sequence of readings x_t, statistical anomaly detection uses hypothesis testing for a given significance level, to determine if a reading is anomalous. The null hypothesis (H_0) is that the dataset contains no anomalies, the alternate hypothesis (H_1) is that the data contains at least one anomaly. Extreme Studentized Deviate (ESD) is such a test where an upper bound is specified as the number of anomalies k, (where $k < 50\%$ of $|x|$), Hochenbaum et al. (2017) recommend setting k as less than 5%. ESD runs the following k times:

$$C_k = \frac{max_k|x_k - \bar{x}|}{s} \tag{6.1}$$

Where $\bar{x}$ and s are the mean and variance of x. This picks k most extreme values and then compares them with the critical value (where n is the number of elements in x):

$$\lambda_k = \frac{(n-k)t_{p,n-k-1}}{\sqrt{(n-k-1+t^2_{p,n-k-1})(n-k+1)}} \tag{6.2}$$

[1]Full details of measures and their parameters are in https://github.com/etsy/skyline/blob/master/src/analyzer/algorithms.py

If this value is anomalous, it is removed from x and repeated k times producing anomalies equal to the largest k such that $C_k > \lambda_k$. A significant drawback of this method is that $\bar{x}$ is sensitive to outliers. The ESD is then modified to use the more robust *Median Absolute Deviation*, the median of the absolute deviations from the median, or MAD = median($|x_i -$ median$(x)|$). The final algorithm is then:

Algorithm 7: Seasonal Hybrid ESD

```
input     : $x$: the values to find anomalies
parameter: $k$ the number of anomalies to find satisfying $k < \frac{1}{2}n$
parameter: $\alpha$ the statistical significance level to accept/reject anomalies
output    : list of anomalies

1  anoms ← empty list;
2  $n$ ← x.length;
3  for $i \leftarrow 0$ to $k$ do
4      ares ← $\frac{\text{median}(x)-x}{\text{MAD}(x)}$;
5      $candAnomalyIndex \leftarrow \text{argmax}(ares)$;
6      delete $x[candAnomalyIndex]$;
7      $p \leftarrow 1 - \frac{\alpha}{2(n-i+1)}$;
8      $t \leftarrow \text{ppf}(p, n-i-1)$ ;
9      $\lambda = \frac{t(n-1)}{\sqrt{(n-i-1+t^2)(n-i+1)}}$;
10     if $max(ares) > \lambda$ then
11         anoms.append($x[candAnomalyIndex]$);
12     end
13 end
14 return $anomalies$
```

Where ppf(q, μ) is the percent point function, which calculates the probability that a random variable from a normal distribution centred at (μ) is above the given percentile (q).

6.4 Stream Outliers for Incident Detection

Based on the literature presented above, there is a gap wherein much research reports the efficacy of algorithms for incident detection trained and evaluated on simulated data. The upside of this is that models can be developed that take in large amounts of data and the effects of simulated incidents can be readily monitored. The drawbacks of this are significant:

- The generated data may be biased to only those types of incidents simulated and the spatiotemporal conditions under which they are conducted.

- Simulations of traffic flow and incidents must be run for every location in order to generate data to train the model.
- Readers are not informed of how these methods perform under real world conditions and incidents.

To this end, a method is preferable that learns to detect incidents from real world data, around the clock, reflecting the types of incidents that actually occur. This research attempts to answer the question "do outliers in traffic flow data indicate the presence of an incident". This is evaluated by using different methods of anomaly detection and evaluating whether or not these anomalies coincide with observed incidents. The following algorithms are selected to detect anomalies in the traffic flow data because of their ability to detect anomalies in seasonal contexts:

- HTM (as described in Chapter 3)
- S-H-ESD (as described in subsubsection 6.3.1.2)

A system was built with the goal of detecting incidents from loop detector data using an algorithm shown to have good properties at anomaly detection on temporal data. It must be stressed that not all anomalous readings necessarily indicate an incident. For example, a Monday public holiday will typically show lower traffic volumes compared to the previous Mondays. To minimise the impact these factors to traffic volume that are not caused by incidents, models inputs must be devised that encode properties about the current date.

There are various kinds of incidents that impact the flow of traffic on a road network that are separate from normal congestion due to increased traffic volume, some examples are:

Obstacles where part of the road becomes blocked off by an obstacle such as crashed vehicles, roadworks, spills, emergency response persons/vehicles.

Disruption where the entire road is blocked off from things such as parades, protests, riots, public events and traffic jams.

6.5 Dataset

The entire dataset was provided by DPTI containing 142,514 accidents in the period 1/1/2006 to 12/2/2015 and is divided into three areas: Adelaide CBD, Adelaide Metropolitan and South Australian Rural. This work only analyses the first two areas, as the majority of traffic flow in those areas is sparse and unrecorded by SCATS. After removing country incidents, there are 137,322 incidents. The provided data has a large number of fields as described in section 2.5, but the ones this work is concerned with are:

- Latitude and longitude of the incident
- Date and time

The traffic data used to infer the occurrence of an incident via its anomalies are the VS and SM datasets as described in section 2.2. In order to reduce the size of the problem of analysing the large amounts of data, this research is only concerned with the last 3 years of incident data in the period 30/1/2012 to 12/2/2015 and traffic data in the period starting 1/1/2012. This should provide adequate time for the algorithms to learn the distribution of the traffic flow data and effectively detect anomalies given that the first incident occurs as 12:40AM on 1/1/2012. During the selected period, there were 299 incidents, 2,836,622 VS records (each record has 24 sensor readings) and 52,281,377 SM readings. The large number of flow readings will also test the ability of the selected methods to process large amounts of data in a workable amount of computation time and memory.

6.6 Method

In order to test the effectiveness of each algorithm for AID, the following test procedure has been devised:

1. For each intersection in a subset of all intersections:
    (a) For each anomaly detection algorithm:
        i. For each dataset in SM and VS for the intersection:
            A. Run the algorithm against and record the time of the anomalies for each strategic input at the intersection (along with any other factors the algorithm outputs regarding the anomaly such as significance) in the database. Each record has the following format:
```
{"intersection": "3001",
"strategic_input": 125,
"algorithm": "HTM",
"ds": "vs",
"datetime": 2012-01-01 14:10:00,
"other":{"field": value}}
```
            Where the `other` field stores information about the anomaly specific to the algorithm such as expected value, significance or anomaly likelihood.
    (b) For each incident in the dataset within 100 metres of a selected intersection:

i. Query the database for anomalies within ±10 minutes of the incident occurring in any of the strategic inputs of the nearest intersection. This should account for incidents that may have inaccuracies in the reported occurrence time or that are dependent on flows from specific movements.
ii. Record the the number of incidents that had associated anomalies as a true positive.
iii. Record those incidents without an associated anomaly as a false negative.
iv. Record those anomalies that do not have an associated incident as false positive.
v. Record those non-anomalous readings without incidents as true negative. Because this number greatly dominates the other cases (most of the time an incident is not occurring and flow is not anomalous), these values are omitted when making a comparison between methods.

The code used to perform this process (along with anomaly detection model parameters and algorithm configuration) is available at https://github.com/JonnoFTW/incident_detection_eval. Due to the sensitive nature of the incident dataset, it is only made available by request the TSC.

The subset of selected intersections are a variety from both CBD and metropolitan locations with varying flow distributions:

- 3001: a CBD location that also includes a tram line
- 3043: a CBD location that connects the CBD with the metropolitan area
- 115 and 100: busy arterial intersections approximately 13km south of the CBD
- 67: a busy intersection just south of the city
- 221: an intersection just south of the city near a street-level tram crossing
- 347: a street-level tram crossing

Given this subset, during the period 1/1/2012 to 12/2/2015 there are:

- 309 crashes within 100 metres of any of the selected intersections.
- 2,897,102 individual VS readings.
- 52,281,377 individual phase count readings from SM data.

The selection of incidents near intersections is not by coincidence, by observing a heatmap (see Figure 6.1) of all accidents in the period: 1/01/2006 to 1/2/2015, it is clear that accidents occur mostly at the intersection of main roads and as such, the incidents selected for detection via anomalous traffic flow should be those causing the most disruption to traffic flow.

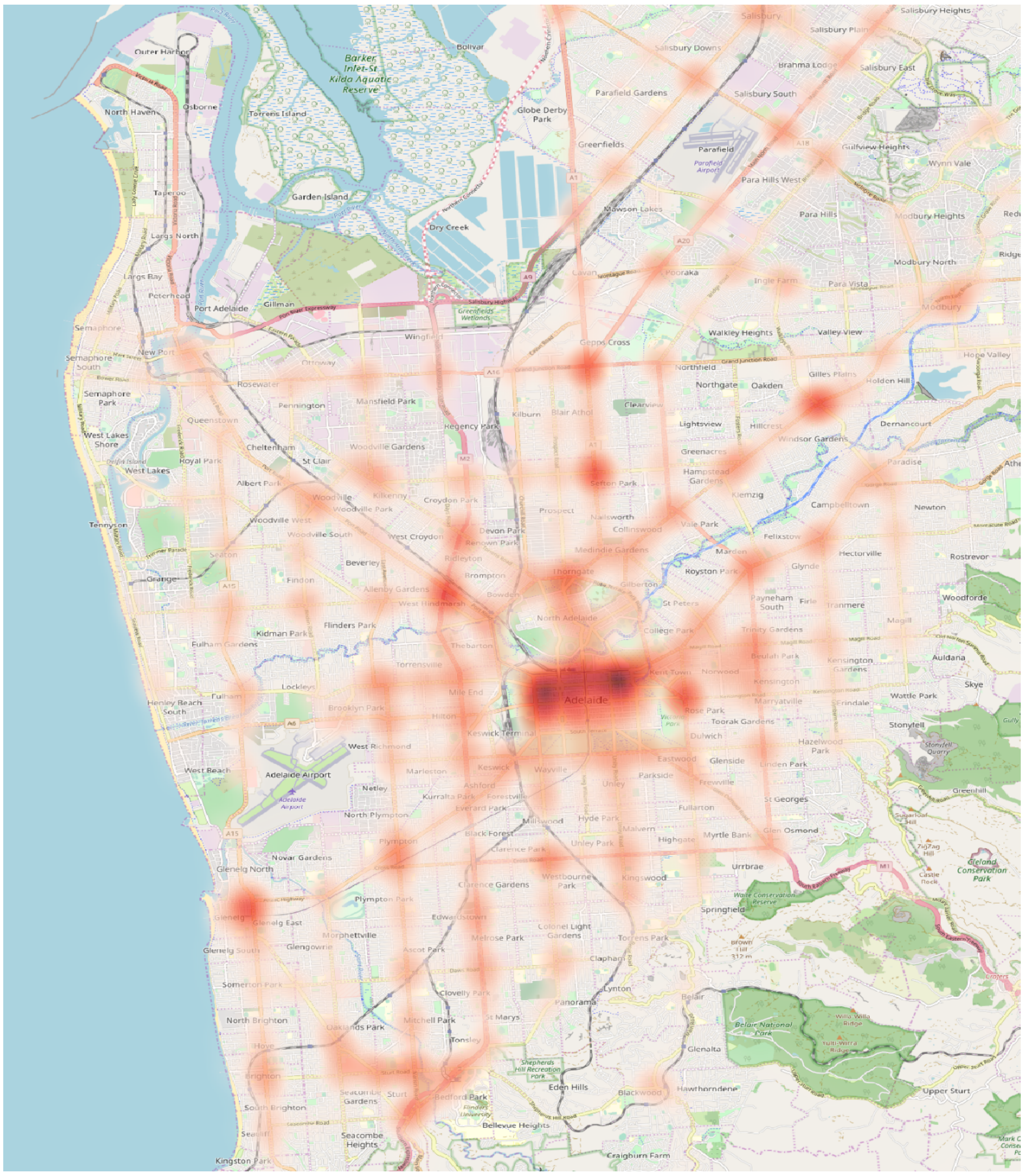
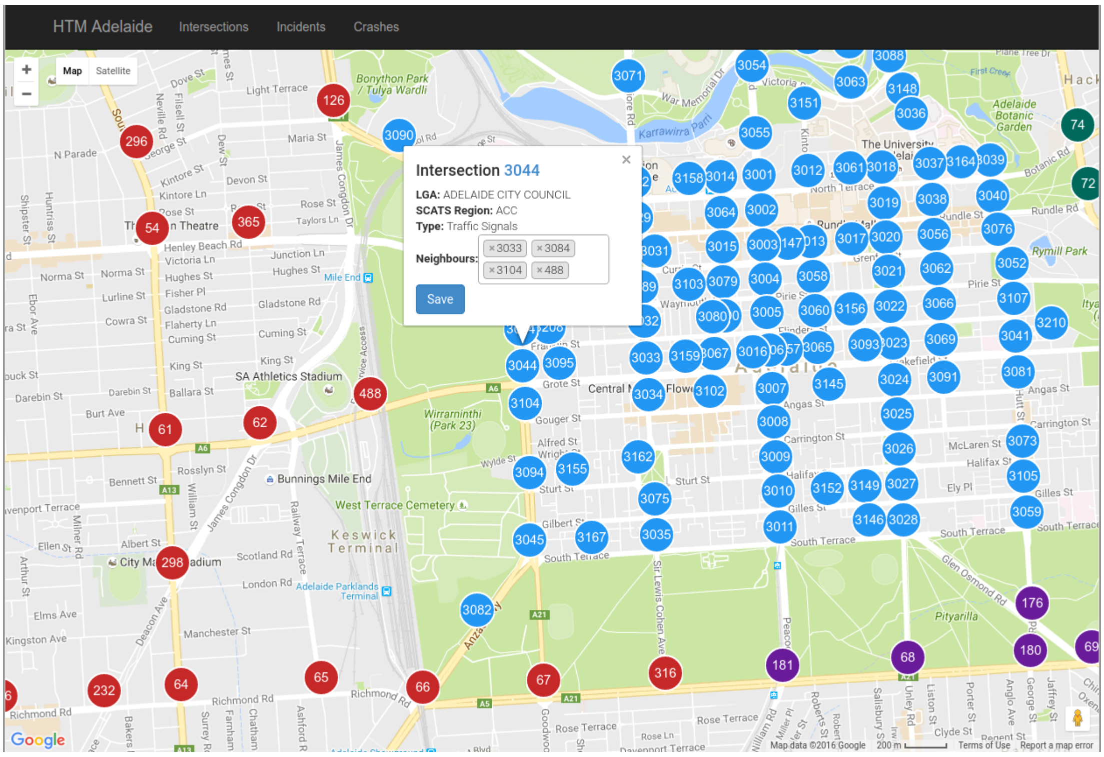


Figure 6.1: Heatmap of accidents (from DPTI crash dataset) in the period 1/1/2006 - 12/12/2015 in Adelaide clustering mostly around intersections of major roads

6.6.1 Interface

As part of the evaluation process, an interface was developed to visualise:

- The layout of signalised intersections in the city, see Figure 6.2
- Information about each signalised intersection, see Figure 6.3:
  - The SCATS diagram
  - The neighbours of the intersection
  - The intersection's strategic input configuration history
  - A plot of traffic flow
- The occurrence of incidents and coincident anomalies, see Figure 6.4



Intersection 3044

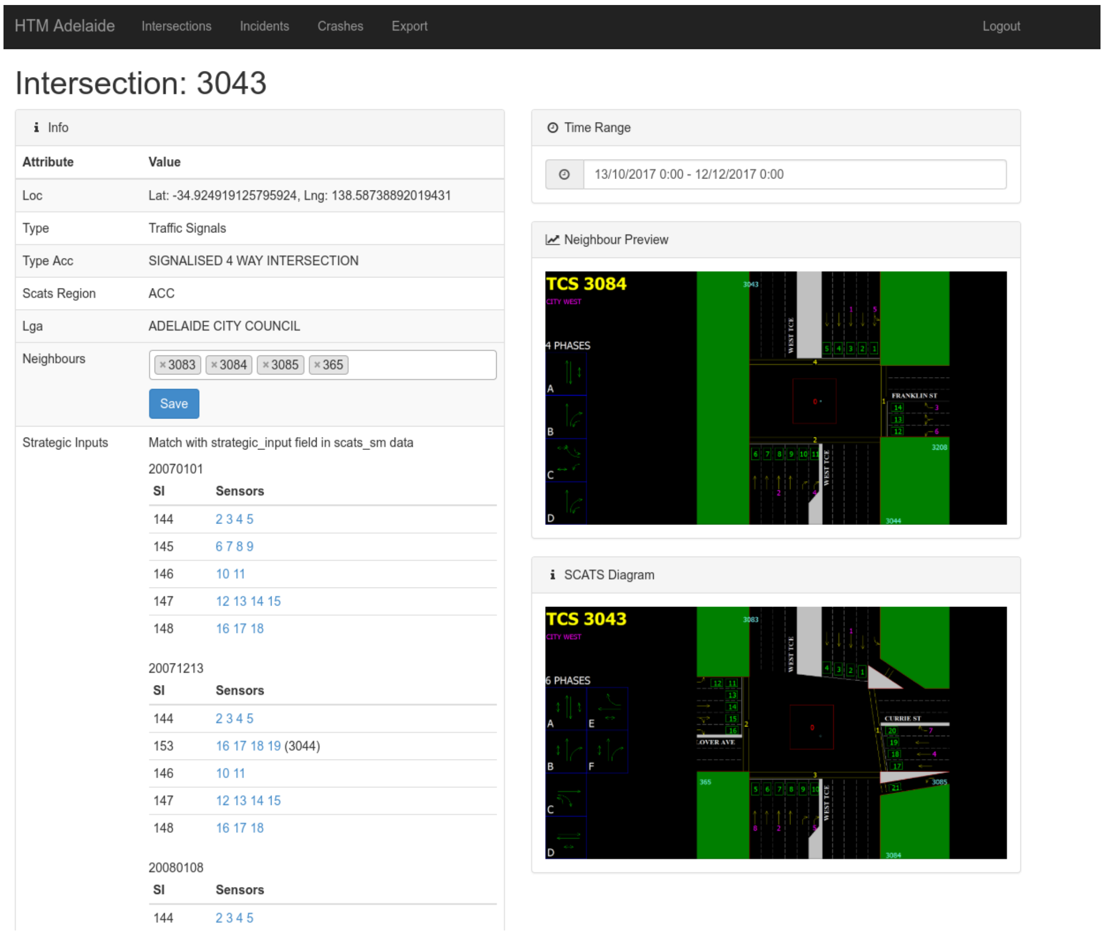
LGA: ADELAIDE CITY COUNCIL
SCATS Region: ACC
Type: Traffic Signals
Neighbours: ×3033 ×3084 ×3104 ×488

Save

Figure 6.2: Map of city shown by the application. Intersections are labelled with their unique identifier and coloured by their SCATS region. Clicking on a particular intersection will show a popup with some details about the intersection and a link to a page with more details

6.7 Results

The scores for HTM and SHESD methods are shown in Table 6.2, given that there are:



Intersection: 3043

Info

Attribute	Value
Loc	Lat: -34.924919125795924, Lng: 138.58738892019431
Type	Traffic Signals
Type Acc	SIGNALISED 4 WAY INTERSECTION
Scats Region	ACC
Lga	ADELAIDE CITY COUNCIL
Neighbours	×3083 ×3084 ×3085 ×365
Strategic Inputs	Match with strategic_input field in scats_sm data

Time Range: 13/10/2017 0:00 - 12/12/2017 0:00

20070101

SI	Sensors
144	2 3 4 5
145	6 7 8 9
146	10 11
147	12 13 14 15
148	16 17 18

20071213

SI	Sensors
144	2 3 4 5
153	16 17 18 19 (3044)
146	10 11
147	12 13 14 15
148	16 17 18

20080108

SI	Sensors
144	2 3 4 5

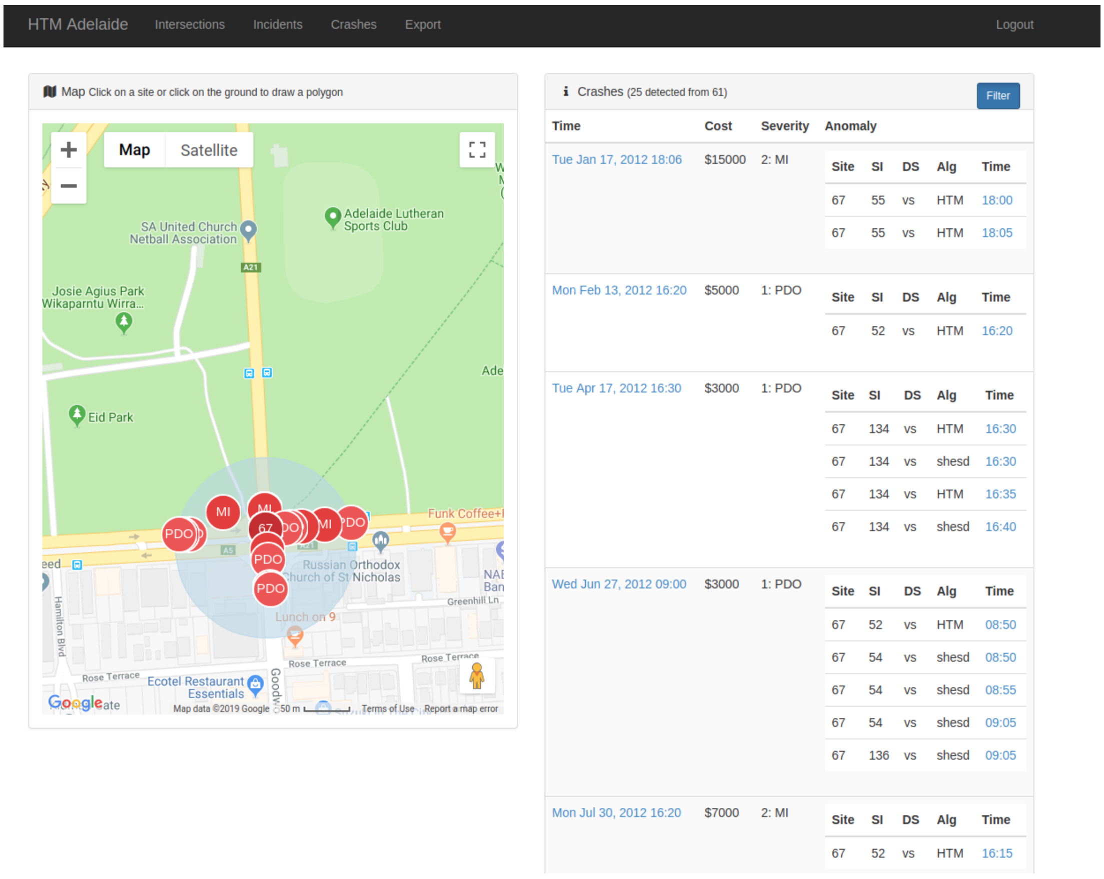
Figure 6.3: Details of the intersection

- 299 incidents within 100m of the selected intersections
- 2,836,622 VS readings
- 52,281,377 SM readings

The standard implementation of SHESD as provided by Hochenbaum et al. (2017) provides two methods for anomaly detection: SHESD TS for timeseries data and SHESD-Vec for other data (ideally temporal sequence data that is missing its timestamp component). Both are evaluated here for completeness, although it is apparent that SHESD TS was not effective at detecting anomalies in either datasets.

While some incidents did coincide with anomalous traffic flow, it is clear here that anomalous traffic flow (as determined by the selected algorithms) does not highly correlate with a crash. Observing Figure 6.6, a few important features can be seen:

- An incident occurred that caused an anomalous drop in traffic around 7am on Tuesday 11 July, 2012. This drop is greater than other drops seen around the same time on other days.


HTM Adelaide
Intersections
Incidents
Crashes
Export
Logout
Map Click on a site or click on the ground to draw a polygon
Map
Satellite
Adelaide Lutheran Sports Club
SA United Church Netball Association
Josie Agius Park Wikaparntu Wirra...
Eid Park
Funk Coffee+
Russian Orthodox Church of St Nicholas
Lunch on 9
Rose Terrace
Ecotel Restaurant Essentials
Hamilton Blvd
Greenhill Ln
MI
PDO
67
Map data ©2019 Google
50 m
Terms of Use
Report a map error
Crashes (25 detected from 61)
Filter

Crashes (25 detected from 61)

Time	Cost	Severity	Anomaly: Site	SI	DS	Alg	Time
Tue Jan 17, 2012 18:06	$15000	2: MI	67	55	vs	HTM	18:00
			67	55	vs	HTM	18:05
Mon Feb 13, 2012 16:20	$5000	1: PDO	67	52	vs	HTM	16:20
Tue Apr 17, 2012 16:30	$3000	1: PDO	67	134	vs	HTM	16:30
			67	134	vs	shesd	16:30
			67	134	vs	HTM	16:35
			67	134	vs	shesd	16:40
Wed Jun 27, 2012 09:00	$3000	1: PDO	67	52	vs	HTM	08:50
			67	54	vs	shesd	08:50
			67	54	vs	shesd	08:55
			67	54	vs	shesd	09:05
			67	136	vs	shesd	09:05
Mon Jul 30, 2012 16:20	$7000	2: MI	67	52	vs	HTM	16:15

Figure 6.4: Accidents at intersection 67 with coincident anomalies. Anomalies are shown along with their damage cost, severity, strategic input, data source, algorithm, time and a link to the view of Figure 6.3 showing flow for the SI around the time of the incident.

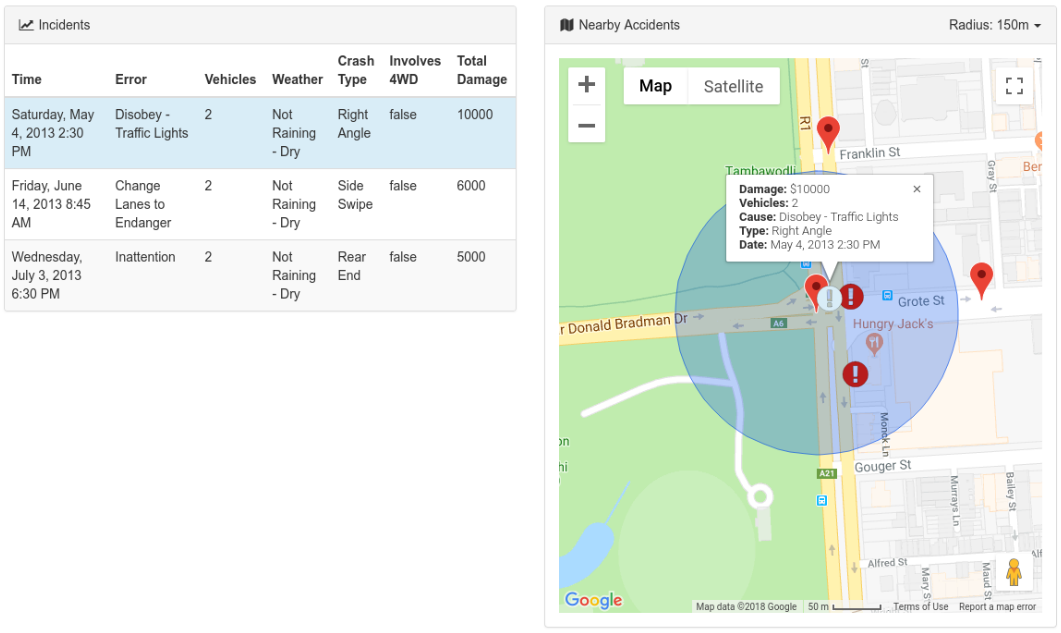
Algorithm	Dataset	True Positive	False Positive	True Negative	False Negative	Anomalies Detected
HTM	VS	43	41447	2795175	256	41490
	SM	41	50091	32650733	258	50132
SHESD Vec	VS	156	143872	2692750	143	144028
	SM	143	238637	32462187	156	238780
SHESD TS	VS	0	756	2835866	299	756
	SM	1	434	32700390	298	435

Table 6.2: Incident detection via anomaly by dataset and algorithm

- Both algorithms regularly find anomalies on this detector, possibly because it is downstream from a level tram crossing. When a tram crosses there is an irregular effect on the flow of traffic, especially during peak hours.

The reasons for these results may be:

- An accident does not cause anomalous traffic flow.



Incidents

Time	Error	Vehicles	Weather	Crash Type	Involves 4WD	Total Damage
Saturday, May 4, 2013 2:30 PM	Disobey - Traffic Lights	2	Not Raining - Dry	Right Angle	false	10000
Friday, June 14, 2013 8:45 AM	Change Lanes to Endanger	2	Not Raining - Dry	Side Swipe	false	6000
Wednesday, July 3, 2013 6:30 PM	Inattention	2	Not Raining - Dry	Rear End	false	5000

Nearby Accidents

Radius: 150m

Damage: $10000
Vehicles: 2
Cause: Disobey - Traffic Lights
Type: Right Angle
Date: May 4, 2013 2:30 PM

Figure 6.5: A map showing accidents for the currently plotted traffic flow

Algorithm	Dataset	Crashes By Intersection							Total
		3001	3043	347	67	115	100	221	Crashes
HTM	VS	12	24	5	23	12	28	7	111
	SM	10	24	6	27	14	19	8	108
SHESD Vec	VS	19	41	9	36	24	45	12	186
	SM	35	49	10	48	33	53	19	247
SHESD TS	VS	1	1	0	0	1	2	0	5
	SM	0	2	0	1	0	0	0	3
Actual		**47**	**54**	**13**	**58**	**41**	**63**	**23**	**299**

Table 6.3: Breakdown of True Positive Incident detections via anomaly for each site, dataset and algorithm

- Anomalous traffic flow does not necessarily indicate an incident.
- Incidents may occur at non-peak hours, such as very early morning, when disruptions to traffic do not cause anomalous flows.
- Some accidents may be at low speed and cleared very quickly, such that vehicles involved may be able to drive off before making an impact on flow.

Although $43/299 \approx 14.4\%$ accidents did coincide with a HTM/VS anomaly, there were 41490 anomalies found meaning that only $43/41490 \approx 0.104$ of anomalies indicate an incident and that the remaining 41447 anomalies were false alarms. This level of false alarms is unacceptably low by most metrics discussed above.

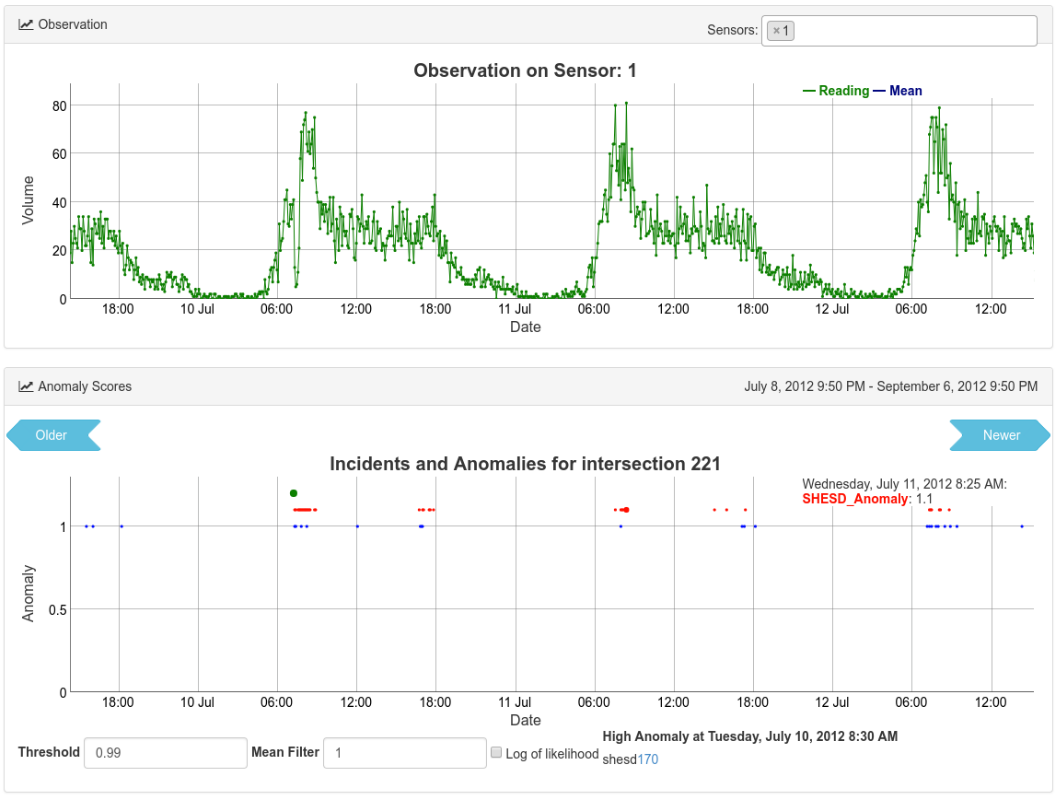


Figure 6.6: Flows near an accident that caused anomalous reduction in traffic flow at strategic input 168. The bottom plot shows incidents in green, HTM anomalies in blue and SHESD anomalies in red. Both algorithms successfully identified this incident as an anomaly. Other anomalies in this particular time period were detected but not coincide with a recorded incident

The coincidence of anomaly and incident for each algorithm and dataset are (excluding SHESD TS):

HTM/VS 0.104%

HTM/SM 0.0818%

SHESD/VS 0.108%

SHESD/SM 0.06%

One reason for this discrepancy may be that the dataset of incidents is dominated by small incidents, where people are rear ended at low speed. Additionally, some incidents may not be detectable by this method at all, since they occur during periods of low traffic flow. The large number of anomalies is likely due to other factors not seen in the existing crash dataset causing anomalous flows.

Table E.1 shows that some incident types do have more anomalies associated with them than others, although this may be due to the small sample size. Future work should examine the entire crash dataset and any anomalies that may be associated with them.

6.8 Conclusion

In this chapter, anomaly detection was evaluated as a novel method for AID. Anomalies were detected using methods that take into account seasonality and their incidence compared with a real-world databases of accidents at selected locations in the Adelaide metropolitan area. While anomalies were found in the data, they far outnumbered accidents, indicating that there are far more anomalies in traffic flow than accidents (at a rate of around 0.1%). While accidents are the cause of some anomalous traffic flows, they are by no means an effective way of indicating incidents in general.

6.9 Future Work

Future work in the area of incident detection should aim to develop models based off real world sensor readings and incident data, with a focus on models that can operate regardless of the time of day or date. Other work may also investigate the nature of anomalous traffic flow and its causes beyond crashes by extending the anomaly detection over the full set of cities. It may become evident that particular kinds of anomalies indicate specific incident types.

Chapter 7

Conclusions and Future Work

In this chapter the main motivations, original contributions, findings and results are summarised and directions for potential future work are described.

The research conducted as part of this thesis was motivated by the potential for intelligent transportation systems, specifically advanced transportation management systems (ATMS) to improve the safety and efficiency of urban road networks. As the population and average wealth of the world increases, so too will the use of road transportation and associated infrastructure. With this comes the inevitable problems and implications of increasingly congested roads: reduced safety, increased travel delay, reduced traveller satisfaction, increased greenhouse gas emissions and reduced productivity. ATMS has a key role to play in bringing order to the otherwise chaotic road environment and improvements and optimisations in this control must be aggressively sought after to improve the safety and efficiency of a foundational part of modern infrastructure.

As the implementation of ITS infrastructure and other data sources such as autonomous vehicles grows, so too will the amount of data produced and stored. Ideally this data would be collected centrally or in a localised swarm of vehicles and used by ITS to make better predictions about network usage, and also used by traffic engineers to assist in system management and the planning of extensions to the existing infrastructure. To this end, algorithms and the hardware they run on such as those showed in previous chapters will need to be extended or developed to accommodate this new influx of data.

The development of traffic control schemes that work in a proactive, rather than reactive manner, based on large volumes of data collected about the road network from a variety of sources will assist in the development of predictive models and decision making systems that ideally exceed the performance of human or current systems. Two tasks that such systems can perform are: short term traffic prediction and automated incident detection. The former seeks to make predictions about the volume of traffic through in an intersections permitted movements into the next phase or in aggregate over a set time period (5 or

15 minutes in SCATS' case). The latter seeks to detect incidents on roads, that is, non-recurrent events that partially or totally inhibit the flow of traffic.

These two issues are actively being researched within in ATMS, as Alsrehin et al. (2019) describes, there continues to be ongoing investigation into the task of traffic flow prediction. The research covered in this thesis makes novel contributions to both tasks, but these topics remain far from closed, as new research will likely integrate the use of external and third party data sources to further improve model performance and ATMS integration.

Literature reviews on short term arterial traffic prediction and arterial incident detection both found two significant problems with much of the existing published work:

1. Experiments are frequently conducted on simulated data, without discussion of their effectiveness or transferability into real world conditions. Such use of simulations can lead to biased models, or the creation of models that cannot be generated or applied to real world data simply because real world data does not exist at the granularity available during simulation.

2. Where real world data is examined, much of it is limited to:

    - Short periods (such as two weeks) of time with fairly typical traffic flows
    - Certain days of the week
    - Selected periods during the day, for example the morning or afternoon peaks of workdays

    Such gaps can make the described methods seem more effective than if they had been analysed on a complete dataset that operated non-stop over the course of several years. This research seeks to rise to the challenge of creating models that can operate regardless of the date, time or location, allowing the investigation of long-term effects of distribution changes on model performance .

To this end, the work described in Chapter 2 contributes a dataset containing a large number of readings from both phase level and aggregated traffic flow SCATS readings in combination with road network and traffic signal configuration information. Such a collection of data and associated software made available within the TSC will allow future research into not only traffic engineering tasks, but wider timeseries and graph data analysis tasks. Unfortunately, due to the privacy demands of SCATS and DPTI, access to this data is not available to the wider research community without special permission.

7.1 Short Term Arterial Traffic Prediction

Short term traffic prediction is the task of predicting future traffic flows through an intersection. Here previous work largely worked on aggregate flows, typically between 3 and 15 minutes, in this research, next phase flows in addition to aggregate data were examined. Flow data in SCATS is collected by loop detectors embedded in the road for each monitored lane at a signalised intersection and typically used by SCATS in order to select cycle plans that define phase durations as defined by a traffic engineer. These sensors can be on freeways, motorways, at various places on arterial roads, and signalised intersection stop lines. These contributions focus on those flows through signalised intersections in an urban setting, predicting flows in the next phase and in aggregated intervals of 5 minutes. This research sets itself apart from previous work by making predictive models that perform online based on years of historical data and evaluating performance over a variety of locations. The intuition being that when models are required to run for years without human intervention, they must adapt to inevitable changes in overall distribution of flow.

Numerous methods were evaluated for short term arterial traffic prediction in a variety of urban locations, from central business district to urban arterial roads. Online learning models had not been explored in the literature of traffic flow prediction, that is, models that continuously update their parameters as new data is received. The advantage being that they can adjust to changes in overall distribution (as opposed to seasonal cycles of arbitrary length), caused by factors such as infrastructure additions or reconfiguration, changes in transport demand, and extended roadworks. This novel application and demonstration of efficacy of online learning is deemed to be a useful contribution to the body of knowledge on this particular task.

HTM has emerged (Hawkins et al. 2016) as a group of algorithms capable of online learning and predicting timeseries data. Inspired by empirical models of the mammalian neocortex (the region of the brain responsible for sensory perception, cognition and motor control), HTM seeks to replicate the behaviour by creating binary encodings of input data and passing them to a layer of columns that learn the spatial layout of inputs and then another layer that learns the temporal layer. This particular model had not been applied to the task of short term traffic prediction before; proving to be effective and competitive with the other state of the art sequence prediction model: LSTM (Lv et al. 2014).

The contributions of this research showed the efficacy of HTM at short term traffic prediction and that hyperparameter optimisation techniques could enable HTM and LSTM, SARIMA models to achieve competitive results in both next-phase and aggregated tasks. Additionally, a novel Markov Model variant (incorporating online learning and a variety of fallback methods) was introduced and shown to outperform all other models in terms of root mean squared error. This result is likely because it reverted to a rolling mean when it could not use a transition probability predict the next step, both methods that made for accurate predictions, although falling back to a rolling mean meant the model did not effectively predict the frequent variation in flow between readings.

Extensions to the contributed Markov Model include:

- Adding a decay factor (similar to that used by HTM, where transitions that are not used have their probability decreased) to the transition probability table.
- Using transition probabilities from a historic period (as opposed to the entire observed history).
- Adding additional hidden features

Future work in this field should investigate the use of feature selection techniques to select the most relevant factors to a particular intersection's future flows (TPE hyperparameter optimisation appears to be a prime candidate method for this). Specifically investigating temporal history from nearby intersections beyond the upstream and downstream readings of previous work. Neural network models could also be investigated that use additional recurrent layer types than used here, namely Gated Recurrent Units (Cho et al. 2014) and Convolutional LSTM (Shi et al. 2015). Models may also be developed that predict multiple steps ahead, beyond the single step predictions performed for next-phase prediction.

Real et al. (2017) showed that genetic algorithms can produce a fully trained neural network that performs with near state-of-the-art performance on a 10 class image classification task with little human intervention (the system started with the simplest possible network and built from there, removing bias from the experimenter about any particular structure). This success can be drawn upon to develop novel neural networks for short term traffic prediction, and potentially other time-series regression tasks.

7.2 Anomalies for Incident Detection

Incidents are those non-recurrent congestion events events that cause partial or complete congestion of roads, limiting the amount of traffic that can flow. These incidents are separate from normal congestion, where a significant enough flow of traffic on a road causes the overall flow of a road to decrease simply due to increased occupation. Non-recurrent congestion is caused by external factors such as illegal parking or breakdown of vehicles, crashes, spilled loads, burst water infrastructure, road or other infrastructure maintenance, and public events such as protests, marches or riots. The detection of these incidents is of vital importance to traffic management as the deployment of services ensures the quick clearance of the incident in order to reduce the duration of the subsequent congestion, delays and secondary incidents.

Automatic Incident Detection (AID) has long been the subject of research, with the majority of previous work focusing on highway traffic where simple algorithms have been devised to detect incidents. A review of existing literature into investigations of AID in arterial roads was often limited to specific types of incidents modelled using simulation software at certain times of the day. The work in this research attempts to identify the presence of accident based incidents in real world from anomalies in flow data and evaluating the performance of this idea by comparing the presence of known accidents against anomalies.

The methods used to detect anomalies in strategic input data (where a strategic input is the collection of sensors used in a particular movement), were HTM and SHESD. The incidence of anomalies in traffic flow data far exceeds the number of accidents, and it was found, that even though anomalies exist in traffic flow in both SM and VS datasets, their incidence did not necessarily coincide with an accident. HTM found 50,132 anomalous datapoints in the SM flow data, 41 of these anomalies were found to have occurred within 10 minutes of a recorded accident out of 299 total accidents. In other words the contribution of this research effort is that accidents do not necessarily cause anomalous traffic flow. This isn't to say that anomalous traffic flow is not of interest, just that it *should not* be used to infer the presence of an incident.

Future work here may involve investigating the actual causes of anomalous traffic flows, creating larger datasets that record the occurrence of real-world vehicle breakdowns, emergency road maintenance, protests etc. so that the effects of all types of incidents can be determined and modelled. As with future work suggested above for short term traffic flow prediction, when developing future algorithms for AID, research should aim to develop models that have been shown to work at all times of the year and all hours of every day, ideally based off real-world data.

7.3 Final Remarks

The datasets and tools generated as part of this work have already enabled work in other fields at Flinders University, specifically, traffic engineering students and researchers now readily access and generate easy to use data files and diagrams for use in their own educational and research projects. It is hoped that the database of VS, SM, crash and location data will be the basis of future research work into traffic behaviour in the Adelaide area.

The author would like to give his sincerest gratitude to Flinders University in providing the space, resources and assistance needed to complete this important research.

Appendix A

Publications, Awards and Software Produced During Candidature

A.1 Publications

The following paper has been accepted for publication (of which the thesis author is the primary author):

- Mackenzie, J., Roddick, J. F., & Zito, R. (2018). An Evaluation of HTM and LSTM for Short-Term Arterial Traffic Flow Prediction. IEEE Transactions on Intelligent Transportation Systems, 99, 1–11. http://doi.org/10.1109/TITS.2018.2843349

A.2 Awards

- Flinders University Research Scholarship (FURS), 2014-2018
- First Prize in *Numenta HTM Challenge 2015* for "HTM for Adelaide Arterial Traffic Flow" https://devpost.com/software/htm-models-adelaide

A.3 Software Projects

The following software projects were completed during the PhD candidature (although not necessarily related to the main research) at Flinders University:

- "HTM Models Adelaide" for viewing and analysing SCATS data with a variety of models https://github.com/JonnoFTW/htm-models-adelaide

- “Webcan” for analysing and reporting vehicle data captured by “rpi-can-logger” https://github.com/JonnoFTW/webcan
- “rpi-can-logger” suite for extracting and parsing CAN data from a range of vehicles including OBD-II compliant cars, Tesla Model X, Mitsubishi Outlander Plugin Hybrid Electric Vehicle, Bustech Prototype Electric Buses, and other FMS equipped buses https://github.com/JonnoFTW/rpi-can-logger
- “TonsleyLEDManager” application for development and display of pixel art animations on the LED display at Tonsley. https://github.com/JonnoFTW/TonsleyLEDManager
- “htm-cl” parallel and performant implementation of HTM in Python and OpenCL https://github.com/JonnoFTW/htm-cl
- “nn-cl” neural network implementation in Python and OpenCL https://github.com/JonnoFTW/nn-cl
- “markov-img-gen” research into using Markov Chains for image/texture and audio generation https://github.com/JonnoFTW/markov-img-gen

Appendix B

Accident Data

The different types of accidents are:

- Hit Parked Vehicle
- Right Angle
- Side Swipe
- Rear End
- Right Turn
- Hit Pedestrian
- Head On
- Roll Over
- Hit Fixed Object
- Other
- Left Road - Out of Control
- Hit Animal
- Hit Object on Road

The causes of accidents are:

- Opening or Closing Door
- Disobey - Traffic Lights
- Fail to Give Way
- Change Lanes to Endanger
- Inattention
- Follow Too Closely
- Fail to Stand

- Overtake Without Due Care
- Fail to Keep Left
- Incorrect Turn
- Disobey - Stop Sign
- Disobey - Give Way Sign
- Reverse Without Due Care
- Fail to Give Way Right
- Vehicle Fault
- D.U.I.
- Dangerous Driving
- Died Sick or Asleep At Wheel
- Excessive Speed
- Brake Failure
- Drunken Pedestrian
- Misjudgement
- Insecure Load
- Other
- Disobey - Police Signal
- Incorrect or No Signal
- Disobey - Railway Signal
- No Errors

The types of weather are:

- Not Raining
- Unknown
- Raining

Appendix C

HTM and LSTM Performance Comparison for Traffic Prediction on TS3044

Table C.1: Hyperparameter Optimisation Search Space Used for Tuning LSTM Network With Optimal Paramaters Based off Minimum GEH

Parameter	**Range**	**Optimal Parameter at k steps**				
		1	3	6	9	12
Hidden Neurons	{128,196,212,230,244,256, 300, 332, 375, 400, 420}	332	256	332	332	332
Batch Size	{96, 105, 128,148,156,164,196}	128	105	105	105	105
Dropout	[0,1)	0.0923	0.0923	0.0042	0.0042	0.0042
Extra Layer	True, False	True	True	True	True	True
Extra Layer Dropout	[0,1)	0.2269	0.001	0.1314	0.1314	0.1314
Extra Layer Neurons	$[0, 1.1) \times$ hidden neurons	269	72	329	329	329

Table C.2: Summary of Results for TS3002 During Period of Varying Distribution at $k = 1$

Model	GEH	RMSE	MAPE
HTM	1.4401	13.5220	38.9275
LSTM-Batch	1.8144	17.9990	50.4074
LSTM-Online	**1.1809**	**10.8066**	**33.6014**

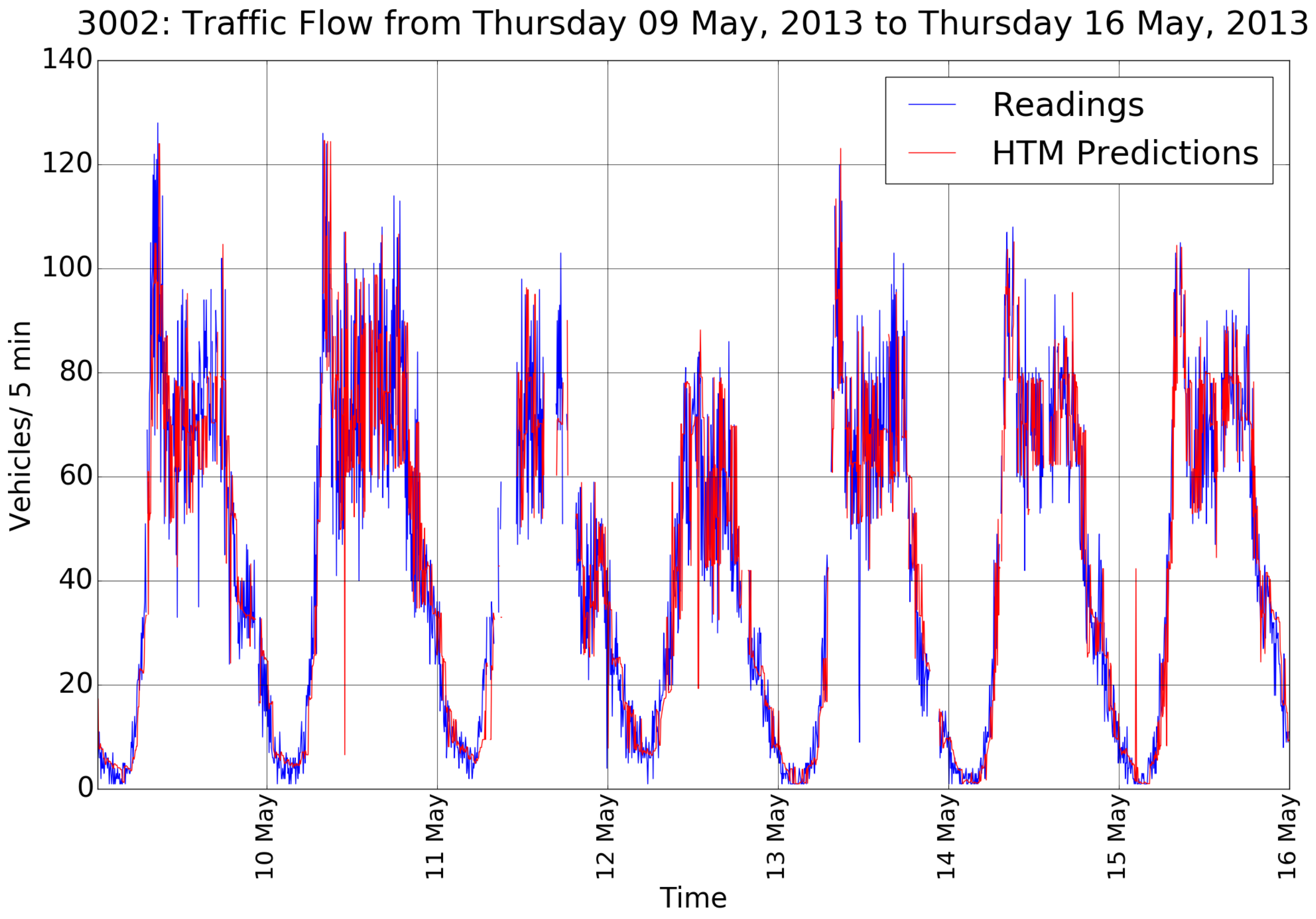


Figure C.1: Traffic Flow Distribution and HTM Predictions

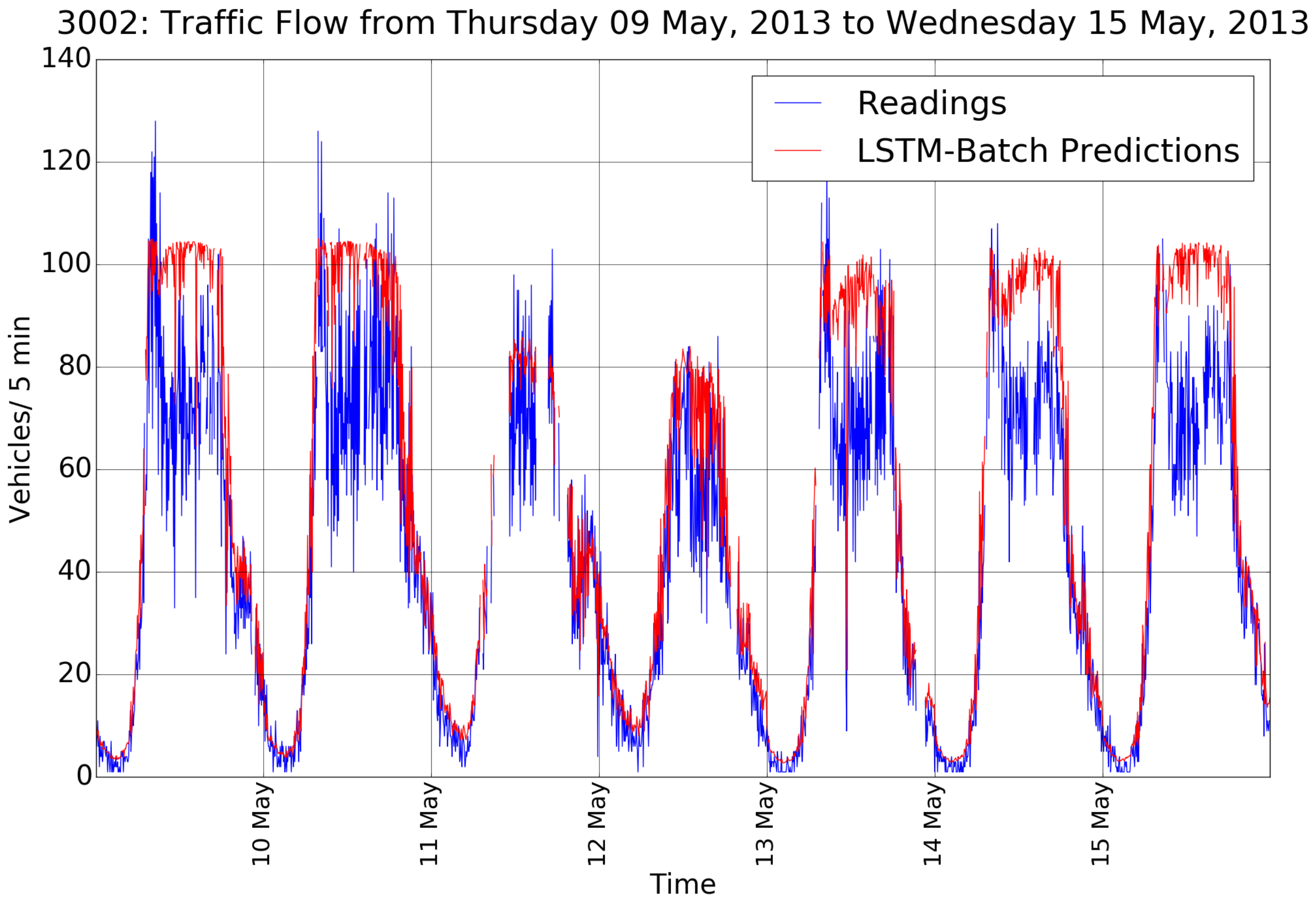


Figure C.2: Traffic Flow Distribution and LSTM Predictions

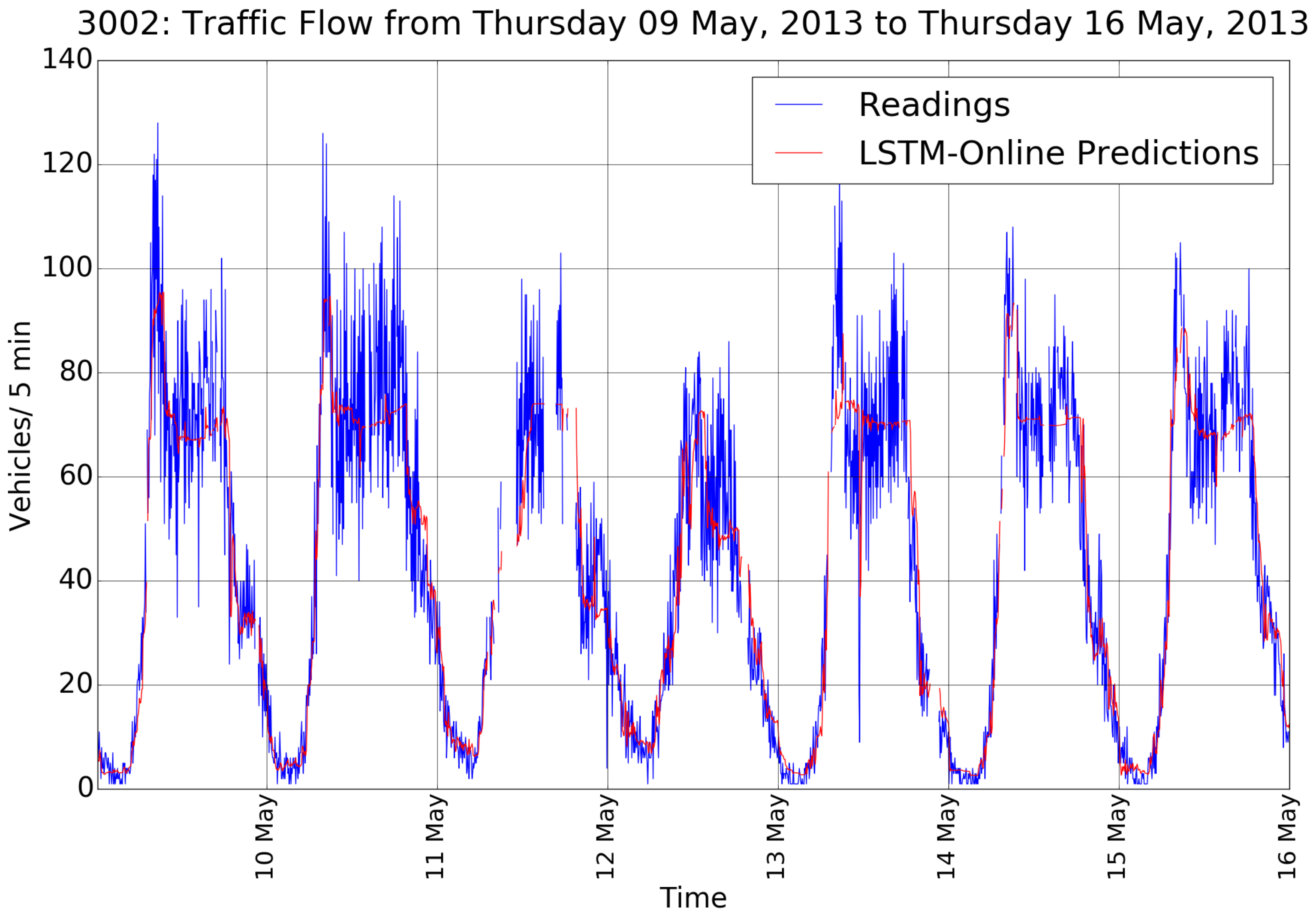


Figure C.3: Traffic Flow Distribution and LSTM-Online Predictions

Appendix D

Hyperparameter Search Space for TS115

The below tables use the following distribution functions as provided by Bergstra et al. (2013):

1. `quniform(min, max, q)` is a quantised uniform distribution over the interval range $[min, max]$ with step q.
2. `uniform(min, max)` is a uniform distribution over the interval $[min, max]$.
3. `choice([a,b,c...])` is a categorical distribution over the given options.

Multiple optimisation jobs were run and some had parameters excluded, these are listed as "unused" where appropriate.

For HTM models, both sets of distributions were used for VS and SM datasets (excluding cycle time from VS as VS does not have a cycle time). Complete parameter files are available online for SM HTM Model (https://github.com/JonnoFTW/htm-models-adelaide/blob/master/engine/sm_model/best_htm.json) and VS HTM Model (https://github.com/JonnoFTW/htm-models-adelaide/blob/master/engine/vs_model/best_htm.json).

Table D.1: Hyperparameter Optimisation Search Space Used for Tuning HTM.

Parameter	**Range**	**Optimal (VS)**	**Optimal (SM)**
activeColumns	quniform(20, 64, 1)	56	47
synPermInactiveDec	uniform(0.0003, 0.1)	0.0632	0.0377
synPermActiveInc	uniform(0.001, 0.1)	0.0836	0.1212
potentialPct	uniform(0.2, 0.85)	0.7573	0.5862
activationThreshold	quniform(5, 20, 1)	8	9
pamLength	quniform(1, 10, 1)	5	7
cellsPerColumn	quniform(8, 32, 2)	14	24
minThreshold	quniform(4, 32, 1)	31	23
alpha	uniform(0.0001, 0.2)	0.04423	0.091
boost	uniform(0.0, 0.1)	0.00722	0.019
tmPermanenceInc	uniform(0.05, 0.2)	0.05875	0.066
maxSynapsesPerSegment	quniform(28, 72, 2)	50	50
newSynapseRatio	uniform(0.4, 0.8)	0.667	0.47
initialPerm	uniform(0.1, 0.33)	0.11619	0.316
maxSegmentsPerCell	quniform(32, 66, 2)	44	40
permanenceDec	uniform(0.01, 0.2)	0.0587	0.192
timeOfDay_width	quniform(16, 201, 2)	54	45
dayOfWeek_width	quniform(20, 201, 2)	132	73
holiday_width	quniform(16,201,2)	133	Unused
dayOfWeek_radius	uniform(6, 15)	11.099	12.44
timeOfDay_radius	uniform(6, 15)	11.389	9.69
weekend_width	quniform(0, 90, 2)	127	70
weekend_radius	quniform(0,90,2)	12.654	1
cycle_time_buckets	quniform(10, 50, 1)	Unused	10

Table D.2: Hyperparameter Optimisation Search Space Used for Tuning LSTM in the VS Task

Parameter	**Range**	**Optimal**
lstm_size_1	quniform(96, 300, 4)	176
lstm_size_2	quniform(96, 300, 4)	88
lstm_size_3	quniform(69, 300, 4)	60
optimizer	choice(['adam', 'rmsprop'])	rmsprop
l1_dropout	uniform(0.001, 0.7)	0.0493
l2_dropout	uniform(0.001, 0.7)	0.6184
l3_dropout	uniform(0.001, 0.7)	0.0120
output_activation	choice(['relu', 'tanh', 'linear'])	relu
state_reset_interval	quniform(1, 100, 1)	Unused
layer_count	quniform(1, 3, 3)	3
l1_reg	uniform(0.0001, 0.1)	Unused
l2_reg	uniform(0.0001, 0.1)	Unused

Table D.3: Hyperparameter Optimisation Search Space Used for Tuning LSTM in the SM Task

Parameter	**Range**	**Optimal**
lstm size 1	quniform(96, 300, 4)	212.0
lstm size 2	quniform(96, 300, 4)	Unused
lstm size 3	quniform(96, 300, 4)	Unused
l1 dropout	uniform(0.001, 0.7)	0.0019
l2 dropout	uniform(0.001, 0.7)	Unused
l3 dropout	uniform(0.001, 0.7)	Unused
layer 1 l1l2 reg	uniform(0.00001, 0.1)	[0.0375, 0.0375]
layer 2 l1l2 reg	uniform(0.00001, 0.1)	Unused
layer 3 l1l2 reg	uniform(0.00001, 0.1)	Unused
optimizer	choice(['adam', 'rmsprop'])	adam
layer count	choice([1,2,3])	1
use weekday	choice([True, False])	False
use minute of day	choice([True, False])	False
use month	choice([True, False])	False
use weekend	choice([True, False])	False
use holidays	choice([True, False])	True
use week number	choice([True, False])	False
use phase time	choice([True, False])	True

Table D.4: TPE Results for SM and VS ARIMA including the selection of exogenous variables

Order	**Range**	**Optimal (SM)**	**Optimal (VS)**
p	quniform(0, 15, 1)	4	15
d	quniform(0, 2, 1)	0	0
q	quniform(0, 5, 1)	3	1
P	quniform(0, 15, 1)	Unused	Unused
D	quniform(0, 2, 1)	Unused	Unused
Q	quniform(0, 10, 1)	Unused	Unused
trend	choice(['n', 'c', 't', 'ct'])	c	c
s	choice([0, 1094, 7919, 288, 1440])	Unused	Unused
use_holidays	choice([True, False])	True	True
use_trend	choice([True, False])	True	True
use_seasonal	choice([True, False])	False	False

D.0.1 Markov Models

The of the grid search over the VS dataset is all combinations of the following:

- Schemes: RAND, MEDIAN, MEAN, LAST
- Bins: $\{10, 15, 20, \ldots, 200\}$
- Order: $\{1, 2, 3, 4, 5, 6, 7\}$

This resulted in the evaluation of 1064 models, so only the best 32 are listed in Table D.5. Full results are available at https://github.com/JonnoFTW/htm-models-adelaide/blob/master/engine/markov_model/mc_115_2_vs_results.txt. Data binning was not used for SM models as the range of observed flow values was smaller.

Table D.5: Grid Search Results for SM Markov Models

Bins	**Order**	**Time (s)**	**Rows**	**Missing**	**Missing %**	**Scheme**	**RMSE**
195	6	23.8081	122643	93600	76.319	MEAN	6.99923
190	6	23.2013	122643	93249	76.033	MEAN	6.99928
195	5	25.1342	122644	80413	65.566	MEAN	7.00572
185	6	24.3707	122643	93124	75.931	MEAN	7.01247
175	6	24.3138	122643	92397	75.338	MEAN	7.01499
180	6	22.7549	122643	92986	75.818	MEAN	7.02108
190	5	23.955	122644	79374	64.719	MEAN	7.02838
170	6	24.6002	122643	91883	74.919	MEAN	7.03385
165	6	22.9402	122643	90535	73.82	MEAN	7.05292
185	5	25.8578	122644	78558	64.054	MEAN	7.06903
160	6	24.689	122643	89199	72.731	MEAN	7.07897
155	6	22.1582	122643	87949	71.711	MEAN	7.11051
180	5	22.492	122644	77717	63.368	MEAN	7.11103
150	6	24.9125	122643	87050	70.978	MEAN	7.13133
175	5	22.9533	122644	76691	62.531	MEAN	7.13167
145	6	24.1353	122643	85559	69.763	MEAN	7.15705
140	6	24.7023	122643	84679	69.045	MEAN	7.177
170	5	25.9684	122644	74415	60.676	MEAN	7.18855
195	6	39.3313	122643	93600	76.319	MEDIAN	7.20709
190	6	35.9802	122643	93249	76.033	MEDIAN	7.21547
185	6	33.8394	122643	93124	75.931	MEDIAN	7.21982
135	6	24.4222	122643	82919	67.61	MEAN	7.2218
180	6	37.2545	122643	92986	75.818	MEDIAN	7.22493
175	6	38.4867	122643	92397	75.338	MEDIAN	7.23096
195	7	24.2653	122642	99806	81.38	MEAN	7.23742
185	7	24.6952	122642	99717	81.307	MEAN	7.24254
175	7	25.0973	122642	98273	80.13	MEAN	7.24326
190	7	22.7966	122642	99728	81.316	MEAN	7.24417
170	6	37.0945	122643	91883	74.919	MEDIAN	7.24765
130	6	24.4451	122643	81415	66.384	MEAN	7.24905
180	7	22.7223	122642	99712	81.303	MEAN	7.25237
170	7	21.858	122642	98214	80.082	MEAN	7.25578

Table D.6: Grid Search Results for VS Markov Models

Order	**Time (s)**	**Rows**	**Missing**	**Missing %**	**Scheme**	**RMSE**
7	77.4235	497650	294962	59.271	MEAN	4.88703
6	93.6856	497651	271721	54.601	MEAN	4.88929
8	84.5633	497649	310698	62.433	MEAN	4.91375
9	87.6242	497648	324706	65.248	MEAN	4.94931
6	114.978	497651	271721	54.601	MEDIAN	5.03245
8	106.013	497649	310698	62.433	MEDIAN	5.05052
5	98.4471	497652	215213	43.246	MEAN	5.06826
7	98.8524	497650	294962	59.271	MEDIAN	5.09192
9	109.846	497648	324706	65.248	MEDIAN	5.12009
5	112.448	497652	215213	43.246	MEDIAN	5.29344
4	102.497	497653	101721	20.44	MEAN	5.71327
4	107.231	497653	101721	20.44	MEDIAN	5.78467
3	104.902	497654	22156	4.452	MEAN	6.59921
5	87.0419	497652	215213	43.246	RAND	6.66108
6	82.4185	497651	271721	54.601	RAND	6.67318
3	105.047	497654	22156	4.452	MEDIAN	6.69884
7	79.5588	497650	294962	59.271	RAND	6.75312
4	93.0203	497653	101721	20.44	RAND	6.8159
8	77.6794	497649	310698	62.433	RAND	6.82689
9	78.1388	497648	324706	65.248	RAND	6.89268
3	101.511	497654	22156	4.452	RAND	7.10937
5	73.6237	497652	215213	43.246	LAST	7.20176
6	69.6085	497651	271721	54.601	LAST	7.20779
7	69.978	497650	294962	59.271	LAST	7.25797
2	103.434	497655	816	0.164	LAST	7.27048
8	67.9915	497649	310698	62.433	LAST	7.29602
4	85.7695	497653	101721	20.44	LAST	7.34066
2	108.423	497655	816	0.164	MEDIAN	7.34255
2	109.12	497655	816	0.164	MEAN	7.36081
9	66.1888	497648	324706	65.248	LAST	7.36119
2	107.49	497655	816	0.164	RAND	7.41978
3	96.8796	497654	22156	4.452	LAST	7.60059
1	120.196	497656	1	0	MEAN	9.97659
1	118.462	497656	1	0	RAND	9.97749
1	114.642	497656	1	0	LAST	9.97791
1	110.951	497656	1	0	MEDIAN	10.0179

Table D.7: Grid Search Results Rolling Mean Model (scores are RMSE)

n	**VS**	**SM**
1	11.765	6.264
2	10.152	5.883
3	9.85	5.689
4	**9.84**	5.588
5	9.995	5.535
6	10.221	5.516
7	10.507	5.496
8	10.852	**5.495**
9	11.222	5.504
10	11.598	5.52
11	11.984	5.534
12	12.375	5.558
13	12.78	5.586
14	13.193	5.617

Appendix E

Incident Detection

Type	**All**	**Detected**	**HTM VS**	**SHESD VS**	**HTM SM**	**SHESD SM**	**Undetected**
Rear End	160	81 (50.6%)	15 (9.4%)	39 (24.4%)	11 (6.9%)	39 (24.4%)	79 (49.4%)
Side Swipe	50	21 (42.0%)	6 (12.0%)	14 (28.0%)	4 (8.0%)	2 (4.0%)	29 (58.0%)
Right Angle	41	15 (36.6%)	2 (4.9%)	11 (26.8%)	1 (2.4%)	8 (19.5%)	26 (63.4%)
Right Turn	26	15 (57.7%)	1 (3.8%)	8 (30.8%)	2 (7.7%)	9 (34.6%)	11 (42.3%)
Hit Fixed Object	7	5 (71.4%)	1 (14.3%)	2 (28.6%)	0 (0.0%)	2 (28.6%)	2 (28.6%)
Roll Over	5	2 (40.0%)	1 (20.0%)	1 (20.0%)	0 (0.0%)	0 (0.0%)	3 (60.0%)
Hit Pedestrian	3	2 (66.7%)	1 (33.3%)	1 (33.3%)	0 (0.0%)	1 (33.3%)	1 (33.3%)
Hit Parked Vehicle	3	1 (33.3%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (33.3%)	2 (66.7%)
Other	2	1 (50.0%)	1 (50.0%)	1 (50.0%)	0 (0.0%)	0 (0.0%)	1 (50.0%)
Hit Animal	1	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)
Head On	1	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (100.0%)
Total	**299**	**143**	**28**	**62**	**28**	**62**	**156**
PDO	210	103 (49.0%)	22 (10.5%)	56 (26.7%)	15 (7.1%)	42 (20.0%)	107 (51.0%)
MI	84	37 (44.0%)	5 (6.0%)	20 (23.8%)	3 (3.6%)	18 (21.4%)	47 (56.0%)
SI	5	3 (60.0%)	1 (20.0%)	1 (20.0%)	0 (0.0%)	2 (40.0%)	2 (40.0%)
Total	**299**	**143**	**28**	**62**	**28**	**62**	**156**

Table E.1: Breakdown of detected incidents for all algorithms by accident type and severity

Bibliography

Abdulhai, B., Ritchie, S. G. & Chair (1999), Towards adaptive incident detection algorithms, *in* 'Proc., 6th World Congress on Intelligent Transport Systems'.
URL: *https://trid.trb.org/view/714257*

Abe, N., Zadrozny, B. & Langford, J. (2006), Outlier detection by active learning, *in* 'Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06', ACM Press, New York, New York, USA, p. 504.
URL: *http://dl.acm.org/citation.cfm?id=1150402.1150459*

Adams, R. P. & MacKay, D. J. C. (2007), 'Bayesian Online Changepoint Detection'.
URL: *https://arxiv.org/abs/0710.3742*

Aggarwal, C. C., Watson, T. J., Ctr, R., Han, J., Wang, J. & Yu, P. S. (2003), 'A Framework for Clustering Evolving Data Streams', *Proceedings of the 29th international conference on Very large data bases* pp. 81–92.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8650*

Aghabayk, K., Moridpour, S., Young, W., Sarvi, M. & Wang, Y.-B. (2011), 'Comparing Heavy Vehicle and Passenger Car Lane-Changing Maneuvers on Arterial Roads and Freeways', *Transportation Research Record: Journal of the Transportation Research Board* **2260**(1), 94–101.
URL: *http://journals.sagepub.com/doi/10.3141/2260-11*

Ahmad, S. & Hawkins, J. (2015), 'Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory'.
URL: *http://arxiv.org/abs/1503.07469*

Ahmad, S. & Hawkins, J. (2016), 'How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites'.
URL: *http://arxiv.org/abs/1601.00720*

Ahmad, S. & Purdy, S. (2016), 'Real-Time Anomaly Detection for Streaming Analytics'.
URL: *http://arxiv.org/abs/1607.02480*

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P.-L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Côté, M.-A., Côté, M., Courville, A., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Kahou, S. E., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I., Graham, M., Gulcehre, C., Hamel, P., Harlouchet, I., Heng, J.-P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrancois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.-A., Mastropietro, O., McGibbon, R. T., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, É., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S. & Zhang, Y. (2016), 'Theano: A Python framework for fast computation of mathematical expressions'.
URL: *http://arxiv.org/abs/1605.02688*

Alsrehin, N. O., Klaib, A. F. & Magableh, A. (2019), 'Intelligent transportation and control systems using data mining and machine learning techniques: A comprehensive study', *IEEE Access* **7**, 49830–49857.

Anbaroglu, B., Heydecker, B. & Cheng, T. (2014), 'Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks', *Transportation Research Part C: Emerging Technologies* **48**, 47–65.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X14002186*

Angeline, P. J., Saunders, G. M. & Pollack, J. B. (1994), 'An evolutionary algorithm that constructs recurrent neural networks', *IEEE transactions on Neural Networks* **5**(1), 54–65.

ARRB (2015), 'Austroads Glossary of Terms'.

Balke, K. N. (1993), An evaluation of existing incident detection algorithms, Technical report, Texas Transportation Institute.
URL: *http://trid.trb.org/view.aspx?id=1170418*

Bell, M. & Thancanamootoo, B. (1988), AUTOMATIC INCIDENT DETECTION WITHIN URBAN TRAFFIC CONTROL SYSTEMS, Technical report,

University of Newcastle upon Tyne.
URL: *http://trid.trb.org/view.aspx?id=263331*

Ben-Akiva, M., Bergman, M. J., Daly, A. J. & Ramaswamy, R. (1984), Modeling inter-urban route choice behaviour, *in* 'Proceedings of the 9th international symposium on transportation and traffic theory', VNU Science Press Utrecht, The Netherlands, pp. 299–330.

Bengio, Y. (2012), 'Practical recommendations for gradient-based training of deep architectures'.
URL: *http://arxiv.org/abs/1206.5533*

Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. (2011), 'Algorithms for Hyper-Parameter Optimization'.
URL: *http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization*

Bergstra, J., Yamins, D. & Cox, D. D. (2013), Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms, Citeseer.

BITRE (2017), Australian Infrastructure Statistics Yearbook 2017, Technical report, Bureau of Infrastucture, Transport and Regional Economics.

Black, J. & Sreedevi, I. (2001), 'Automatic incident detection algorithms', *ITS Decision Database in PATH* .

Bonnefon, J.-F., Shariff, A. & Rahwan, I. (2016), 'The social dilemma of autonomous vehicles', *Science* **352**(6293), 1573 LP – 1576.
URL: *http://science.sciencemag.org/content/352/6293/1573.abstract*

Brandl, O. (2016), 'V2x traffic management', *e & i Elektrotechnik und Informationstechnik* **133**(7), 353–355.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (2000), 'LOF: Identifying Density-Based Local Outliers', *ACM SIGMOD Record* **29**(2), 93–104.
URL: *http://dl.acm.org/citation.cfm?id=335191.335388*

Burge, C. B. & Karlin, S. (1998), 'Finding the genes in genomic DNA', *Current Opinion in Structural Biology* **8**(3), 346–354.
URL: *https://www.sciencedirect.com/science/article/pii/S0959440X98800699*

Byrne, F. (2015), 'Encoding Reality: Prediction-Assisted Cortical Learning Algorithm in Hierarchical Temporal Memory'.
URL: *http://arxiv.org/abs/1509.08255*

Cao, F., Ester, M., Qian, W. & Zhou, A. (2006), 'Density-based clustering over an evolving data stream with noise', *Proceedings of the Sixth SIAM International*

Conference on Data Mining pp. 328–339.
URL: *https://archive.siam.org/meetings/sdm06/proceedings/030caof.pdf*

Chan, K. Y., Dillon, T. S., Singh, J. & Chang, E. (2012), 'Neural-Network-Based Models for Short-Term Traffic Flow Forecasting Using a Hybrid Exponential Smoothing and Levenberg–Marquardt Algorithm', *IEEE Transactions on Intelligent Transportation Systems* **13**(2), 644–654.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6088012*

Chassiakos, A. P. & Stephanedes, Y. J. (1993), 'Smoothing algorithms for incident detection', (1394).
URL: *https://www.researchgate.net/publication/240247850_Smoothing_algorithms_for_incident_detection*

Chen, S., Wang, W., Qu, G. & Lu, J. (2007), Application of Neural Network Ensembles to Incident Detection, *in* '2007 IEEE International Conference on Integration Technology', IEEE, pp. 388–393.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4290502*

Chen, Y., Mittal, A. & Mahmassani, H. S. (2017), Twitter or chatter? involving social media data analysis in traffic incident management, Technical report.

Chen, Y. & Tu, L. (2007), 'Density-based clustering for real-time stream data', *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining KDD 07* **d**, 133.
URL: *http://portal.acm.org/citation.cfm?doid=1281192.1281210*

Cheu, R. L. & Ritchie, S. G. (1995), 'Automated detection of lane-blocking freeway incidents using artificial neural networks', *Transportation Research Part C: Emerging Technologies* **3**(6), 371–388.
URL: *http://www.sciencedirect.com/science/article/pii/0968090X9500016C*

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), 'Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation'.
URL: *http://arxiv.org/abs/1406.1078*

Chollet, F. (2015), 'Keras', https://github.com/fchollet/keras.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B. & LeCun, Y. (2014), 'The Loss Surfaces of Multilayer Networks'.
URL: *http://arxiv.org/abs/1412.0233*

Clark, S. (2003), 'Traffic Prediction Using Multivariate Nonparametric Regression', *Journal of Transportation Engineering* **129**(2), 161–168.
URL: *http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-947X(2003)129:2(161)*

Clement, F. S. C., Vashistha, A. & Rane, M. E. (2015), Driver fatigue detection system, *in* 'Information Processing (ICIP), 2015 International Conference on', IEEE, pp. 229–234.

Coconea, L. & Bellini, E. (2019), 'Advanced traffic management systems supporting resilient smart cities', *Transportation Research Procedia* **41**, 556–558.

Core Data (2016), The Australian Commuting Survey - October 2016, Technical report, Core Data, Sydney.
URL: *https://www.realinsurance.com.au/RealInsurance/media/documents/resources/2016-coredata-australian-commuting-survey.pdf*

Cox, J. (2013), Development of a permanent system to record and analyse Bluetooth travel time and SCATS lane count data, *in* 'Australian Institute of Traffic Planning and Management (AITPM) National Conference, 2013, Perth, Western Australia'.
URL: *http://trid.trb.org/view.aspx?id=1262365*

Cox, J. (2014), 'Adelaide's Bluetooth Travel Time System'.

Cui, Y., Ahmad, S. & Hawkins, J. (2015), 'Continuous online sequence learning with an unsupervised neural network model'.
URL: *http://arxiv.org/abs/1512.05463*

de Souza, A. M., Brennand, C. A., Yokoyama, R. S., Donato, E. A., Madeira, E. R. & Villas, L. A. (2017), 'Traffic management systems: A classification, review, challenges, and future perspectives', *International Journal of Distributed Sensor Networks* **13**(4), 1550147716683612.
URL: *https://journals.sagepub.com/doi/full/10.1177/1550147716683612*

de Souza, A. M., da Fonseca, N. L. S. & Villas, L. (2017), A fully-distributed advanced traffic management system based on opportunistic content sharing, *in* '2017 IEEE International Conference on Communications (ICC)', pp. 1–6.

Deng, J., Dong, W., Socher, R., Li, L., Li, K. & Fei-Fei, L. (2009), ImageNet: A large-scale hierarchical image database, *in* '2009 IEEE Conference on Computer Vision and Pattern Recognition', pp. 248–255.

Devi, Y. U. & Rukmini, M. S. S. (2016), Iot in connected vehicles: Challenges and issues — a review, *in* '2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)', pp. 1864–1867.

Dimitrakopoulos, G. & Demestichas, P. (2010), 'Intelligent Transportation Systems', *IEEE Vehicular Technology Magazine* **5**(1), 77–84.
URL: *http://ieeexplore.ieee.org/document/5430544/*

DTEI (2010), 'AIMSUN Model Development Manual'.

Dukkipati, C., Zhang, Y. & Cheng, L. C. (2018), Lstm based multiple squashing functions deep learning model for advanced traffic management system attack detection, *in* '2018 IEEE 14th International Conference on Control and Automation (ICCA)', pp. 417–422.

Etsy (2015), 'skyline'.
URL: *https://github.com/etsy/skyline*

Farradyne, P. B. (2000), 'Traffic incident management handbook'.

Feldman, O. (2012), The GEH Measure and Quality of the Highway Assignment Models, European Transport Conference, Glasgow, Scotland.

Fu, K., Nune, R. & Tao, J. X. (2015), Social media data analysis for traffic incident detection and management, Technical report.

Gang, X., Kang, W., Wang, F., Zhu, F., Lv, Y., Dong, X., Riekki, J. & Pirttikangas, S. (2015), Continuous Travel Time Prediction for Transit Signal Priority Based on a Deep Network, *in* '2015 IEEE 18th International Conference on Intelligent Transportation Systems', IEEE, pp. 523–528.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7313184*

Gettman, D., Toppen, A., Hales, K., Voss, A., Engel, S. & El Azhari, D. (n.d.), 'Integrating Emerging Data Sources into Operational Practice : Opportunities for Integration of Emerging Data for Traffic Management and TMCs.'.
URL: *https://rosap.ntl.bts.gov/view/dot/34175*

Glorot, X. & Bengio, Y. (2010), 'Understanding the difficulty of training deep feedforward neural networks'.
URL: *http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf*

Goldberg, D. E. & Deb, K. (1991), 'A Comparative Analysis of Selection Schemes Used in Genetic Algorithms', *Foundations of Genetic Algorithms* **1**, 69–93.
URL: *https://www.sciencedirect.com/science/article/pii/B9780080506845500082*

Goldstein, D. (2016), 'Autonomous Vehicles Will Drive Themselves-But They Won't Regulate Themselves', *Hastings Bus. LJ* **13**, 241.

Goplan, S. (2018), 'Legal lessons for Australia from Uber's self-driving car fatality'.

URL: *http://theconversation.com/legal-lessons-for-australia-from-ubers-self-driving-car-fatality-93649*

Hamed, M. M., Al-Masaeid, H. R. & Said, Z. M. B. (1995), 'Short-Term Prediction of Traffic Volume in Urban Arterials', *Journal of Transportation Engineering* **121**(3), 249–254.
URL: *https://doi.org/10.1061/(ASCE)0733-947X(1995)121:3(249)*

Han, L. D. & May, A. D. (1989), Automatic Detection of Traffic Operational Problem on Urban Arterials, Technical Report 8, Institute of Transportation Studies, University of California, Davis.
URL: *http://trid.trb.org/view.aspx?id=318298*

Hawkins, J. & Ahmad, S. (2016), 'Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex', *Frontiers in Neural Circuits* **10**.
URL: *http://arxiv.org/abs/1511.00083*

Hawkins, J., Ahmad, S., Purdy, S. & Lavin, A. (2016), *Biological and Machine Intelligence (BAMI)*, 0.4 edn.
URL: *http://numenta.com/biological-and-machine-intelligence/*

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep Residual Learning for Image Recognition, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Henderson, J., Salzberg, S. & Fasman, K. H. (1997), 'Finding Genes in DNA with a Hidden Markov Model', *Journal of Computational Biology* **4**(2), 127–141.
URL: *http://www.liebertpub.com/doi/10.1089/cmb.1997.4.127*

Hochenbaum, J., Vallis, O. S. & Kejariwal, A. (2017), 'Automatic Anomaly Detection in the Cloud Via Statistical Learning'.
URL: *http://arxiv.org/abs/1704.07706*

Hochreiter, S. & Schmidhuber, J. (1997), 'Long Short-Term Memory', *Neural Computation* **9**(8), 1735–1780.
URL: *http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735*

Howard, D. & Dai, D. (2014), Public perceptions of self-driving cars: The case of Berkeley, California, *in* 'Transportation Research Board 93rd Annual Meeting', Vol. 14.

Huang, W., Song, G., Hong, H. & Xie, K. (2014), 'Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning', *IEEE Transactions on Intelligent Transportation Systems* **15**(5), 2191–2201.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6786503*

Hyndman, R. J. & Athanasopoulos, G. (2018), *Forecasting: principles and practice*, OTexts.

Ivan, J. N. (1997), 'Neural network representations for arterial street incident detection data fusion', *Transportation Research Part C: Emerging Technologies* **5**(3-4), 245–254.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X97000181*

Ivan, J. N., Schofer, J. L., Koppelman, F. S. & Massone, L. L. E. (1995), 'Real-time data fusion for arterial street incident detection using neural networks', *Transportation research record* (1497), 27–35.
URL: *https://trid.trb.org/view/452678*

Jain, A., Zhang, Z. & Chang, E. Y. (2006), Adaptive non-linear clustering in data streams, *in* 'Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06', ACM Press, New York, New York, USA, p. 122.
URL: *http://dl.acm.org/citation.cfm?id=1183614.1183636*

Jain, L. C., Seera, M., Lim, C. P. & Balasubramaniam, P. (2014), 'A review of online learning in supervised neural networks', *Neural Computing and Applications* **25**(3-4), 491–509.
URL: *http://link.springer.com/10.1007/s00521-013-1534-4*

Josephine Lim (2015), 'Police shocked by wild daylight chase on Southern Expressway which reached 160 km/h — Adelaide Now'.
URL: *https://www.adelaidenow.com.au/news/south-australia/police-chase-on-southern-expressway-ends-with-car-in-flames-at-ohalloran-hill/news-story/164a02748e4ea95b25db7145a3e06758*

Jozefowicz, R., Zaremba, W. & Sutskever, I. (2015), An empirical exploration of recurrent network architectures, *in* 'Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37', JMLR.org, pp. 2342–2350.
URL: *https://dl.acm.org/citation.cfm?id=3045367*

Karagiannidis, G. & Lioumpas, A. (2007), 'An Improved Approximation for the Gaussian Q-Function', *IEEE Communications Letters* **11**(8), 644–646.
URL: *http://ieeexplore.ieee.org/document/4289989/*

Khan, S. I. & Ritchie, S. G. (1998), 'Statistical and neural classifiers to detect traffic operational problems on urban arterials', *Transportation Research Part C: Emerging Technologies* **6**(5-6), 291–314.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X99000054*

Khanal, M. (2012), 'How Rapidly should Developing Countries Implement Intelligent Transportation Systems (ITS) to Solve the Growing Urban Traffic Congestion Problem?'.
URL: *https://www.omicsonline.org/open-access/how-rapidly-should-developing-countries-implement-intelligent-transportation-systems-its-to-solve-the-growing-urban-traffic-congestion-problem-2165-784X.1000e106.php?aid=6900*

Kharas, H. (2010), 'The Emerging Middle Class in Developing Countries', (285).
URL: *https://www.oecd-ilibrary.org/content/paper/5kmmp8lncrns-en*

Kingma, D. & Ba, J. (2014), 'Adam: A Method for Stochastic Optimization'.
URL: *http://arxiv.org/abs/1412.6980*

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by Simulated Annealing', *Science* **220**(4598), 671 LP – 680.
URL: *http://science.sciencemag.org/content/220/4598/671.abstract*

Klein, L. A., Mills, M. K. & Gibson, D. R. P. (2006), Traffic Detector Handbook, Technical Report October, Federal Highway Administration.
URL: *http://trid.trb.org/view.aspx?id=349412*

Krizhevsky, A. & Hinton, G. (2009), Learning multiple layers of features from tiny images, Technical report.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, *in* F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems 25', Curran Associates, Inc., pp. 1097–1105.
URL: *http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf*

Kulp, D., Haussler, D., Reese, M. G. & Eeckman, F. H. (1996), 'A generalized hidden Markov model for the recognition of human genes in DNA.', *Proceedings. International Conference on Intelligent Systems for Molecular Biology* **4**, 134–42.
URL: *http://www.ncbi.nlm.nih.gov/pubmed/8877513*

Lakshminarasimhan, M. (2016), 'Iot based traffic management system'.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.
URL: *http://dx.doi.org/10.1038/nature14539*

Lee, W.-H., Tseng, S.-S., Shieh, J.-L. & Chen, H.-H. (2011), 'Discovering Traffic Bottlenecks in an Urban Network by Spatiotemporal Data Mining on Location-Based Services', *IEEE Transactions on Intelligent Transportation*

Systems **12**(4), 1047–1056.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5766752*

Leith, P. (2010), 'The Rise and Fall of the Legal Expert System', *European Journal of Law and Technology* **1**(1).
URL: *http://ejlt.org/article/view/14*

Levinson, S. E., Rabiner, L. R. & Sondhi, M. M. (1986), 'Hidden Markov model speech recognition arrangement'.

Lewis, J. P. (1995), Fast template matching, *in* 'Vision interface', Vol. 95, pp. 15–19.

Lewis, R. (2007), 'Metaheuristics can solve sudoku puzzles', *Journal of heuristics* **13**(4), 387–401.

Lo Bosco, G. & Di Gangi, M. (2017), Deep Learning Architectures for DNA Sequence Classification, *in* 'Lecture Notes in Computer Science', Vol. 10147, pp. 162–171.

Long, J., Shelhamer, E. & Darrell, T. (2015), Fully Convolutional Networks for Semantic Segmentation, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Lourakis, M. I. A. (2005), 'A Brief Description of the Levenberg-Marquardt Algorithm Implemened by levmar'.

Lv, Y., Duan, Y., Kang, W., Li, Z. & Wang, F.-Y. (2014), 'Traffic Flow Prediction With Big Data: A Deep Learning Approach', *IEEE Transactions on Intelligent Transportation Systems* **16**(2), 1–9.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6894591*

Mackenzie, J., Roddick, J. F. & Zito, R. (2018), 'An Evaluation of HTM and LSTM for Short-Term Arterial Traffic Flow Prediction', *IEEE Transactions on Intelligent Transportation Systems* **99**, 1–11.

Mahmassani, H., Haas, C., Zhou, S. & Peterman, J. (1998), 'Evaluation of incident detection methodologies', *Work* .
URL: *https://www.researchgate.net/publication/237786152_Evaluation_of_Incident_Detection_Methodologies*

Michau, G., Nantes, A., Chung, E., Abry, P. & Borgnat, P. (2014), 'Retrieving dynamic origin-destination matrices from Bluetooth data'.
URL: *http://eprints.qut.edu.au/66511/1/Michau_et_al_2014_Retrieving_Dynamic_Origin-Destination_Matrices_from_Bluetooth_Data.pdf*

MongoDB (2018), 'BSON Types — MongoDB Manual 3.6'.
URL: *https://docs.mongodb.com/manual/reference/bson-types/*

Mountcastle, V. (1997), 'The columnar organization of the neocortex', *Brain* **120**(4), 701–722.
URL: *http://brain.oxfordjournals.org/content/120/4/701.short*

Nellore, K. & Hancke, G. P. (2016), 'A survey on urban traffic management system using wireless sensor networks', *Sensors* **16**(2).
URL: *https://www.mdpi.com/1424-8220/16/2/157*

Norley, K. (2011), Urban rail infrastructure – the path from comprehensive transport plans to the recent experience, *in* 'Australasian Transport Research Forum (ATRF), 34th, 2011, Adelaide, South Australia, Australia', p. 15.
URL: *http://www.worldtransitresearch.info/research/4352*

Oh, K.-S. & Jung, K. (2004), 'GPU implementation of neural networks', *Pattern Recognition* **37**(6), 1311–1314.
URL: *https://www.sciencedirect.com/science/article/pii/S0031320304000524*

Olah, C. (2015), 'Understanding LSTM Networks – colah's blog'.
URL: *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

OpenCV (2015), 'Open Source Computer Vision Library', https://github.com/opencv/opencv.

Pack, M. L., Ivanov, N., Bauer, J. & Birriel, E. (n.d.), 'Considerations of Current and Emerging Transportation Management Center Data'.
URL: *https://rosap.ntl.bts.gov/view/dot/43582*

Parkany, E. & Hall, M. (2005), 'A Complete Review of Incident Detection Algorithms & Their Deployment : What Works and What Doesn't', (00).

Payne, H. J. & Tignor, S. C. (1978), 'Freeway incident-detection algorithms based on decision trees with states', *Transportation Research Record* (682).

Persaud, B. N., Hall, F. L. & Hall, L. M. (1990), 'CONGESTION IDENTIFICATION ASPECTS OF THE MCMASTER INCIDENT DETECTION ALGORITHM', *Transportation Research Record* (1287).
URL: *https://trid.trb.org/view.aspx?id=352877*

Peterman, J. (1999), 'Calibration and evaluation of automatic incident detection algorithms', *University of Texas, Austin, TX. Master's Thesis* .

Pokrajac, D., Lazarevic, a. & Latecki, L. (2007), 'Incremental Local Outlier Detection for Data Streams', *2007 IEEE Symposium on Computational Intelligence and Data Mining* (Cidm), 504–515.

URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4221341*

Rabiner, L. R. (1989), 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proceedings of the IEEE* **77**(2), 257–286.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Le, Q. V. & Kurakin, A. (2017), 'Large-Scale Evolution of Image Classifiers', *CoRR* **abs/1703.0**.
URL: *http://arxiv.org/abs/1703.01041*

Rizzo, R., Fiannaca, A., La Rosa, M. & Urso, A. (2016), Classification Experiments of DNA Sequences by Using a Deep Neural Network and Chaos Game Representation, *in* 'Proceedings of the 17th International Conference on Computer Systems and Technologies 2016', CompSysTech '16, ACM, New York, NY, USA, pp. 222–228.
URL: *http://doi.acm.org/10.1145/2983468.2983489*

Robinson, S. & Polak, J. W. (2006), Inductive Loop Detector Data Cleaning Treatments and Their Effect on Performance of Urban Link Travel Time Models, *in* 'Transportation Research Board 85th Annual Meeting'.
URL: *http://trid.trb.org/view.aspx?id=776647*

RTANSW (2004), 'SCATS 6.4.1 Operating Instructions'.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**(6088), 533–536.
URL: *http://www.nature.com/doifinder/10.1038/323533a0*

Salzberg, S. L., Delcher, A. L., Kasif, S. & White, O. (1998), 'Microbial gene identification using interpolated Markov models', *Nucleic Acids Research* **26**(2), 544–548.
URL: *https://dx.doi.org/10.1093/nar/26.2.544*

Santos, G. (2017), 'Road transport and CO2 emissions: What are the challenges?', *Transport Policy* **59**, 71–74.
URL: *https://www.sciencedirect.com/science/article/pii/S0967070X17304262*

Shahsavari, B. & Abbeel, P. (2015), Short-Term Traffic Forecasting: Modeling and Learning Spatio-Temporal Relations in Transportation Networks Using Graph Neural Networks, PhD thesis, University of California, Berkeley.
URL: *http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-243.html*

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k. & Woo, W.-c. (2015), 'Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting'.
URL: *http://arxiv.org/abs/1506.04214*

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., de Carvalho, A. C. P. L. F. & Gama, J. (2013), 'Data Stream Clustering: A survey', *ACM Computing Surveys* **46**(1), 1–31.
URL: *http://dl.acm.org/citation.cfm?id=2522968.2522981*

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Šingliar, T. & Hauskrecht, M. (2010), 'Learning to detect incidents from noisily labeled data', *Machine Learning* **79**(3), 335–354.

Smarter and more connected: Future intelligent transportation system (2018), *IATSS Research* **42**(2), 67 – 71.

Smith, B. L. & Demetsky, M. J. (1994), 'SHORT-TERM TRAFFIC FLOW PREDICTION: NEURAL NETWORK APPROACH', *Transportation Research Record* (1453).
URL: *http://trid.trb.org/view.aspx?id=424677*

Smith, R. (2007), An Overview of the Tesseract OCR Engine, *in* 'Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)', pp. 629–633.
URL: *https://research.google.com/pubs/pub33418.html*

Stathopoulos, A. & Karlaftis, M. (2001), 'Temporal and Spatial Variations of Real-Time Traffic Data in Urban Areas', *Transportation Research Record* **1768**(16), 135–140.

Stathopoulos, A. & Karlaftis, M. G. (2003), 'A multivariate state space approach for urban traffic flow modeling and prediction', *Transportation Research Part C: Emerging Technologies* **11**(2), 121–135.
URL: *https://www.sciencedirect.com/science/article/pii/S0968090X03000044*

Stephanedes, Y. J. & Chassiakos, A. P. (n.d.), 'Application of Filtering Techniques for Incident Detection', *Journal of Transportation Engineering* (1), 13–26.
URL: *https://www.researchgate.net/publication/245306239_Application_of_Filtering_Techniques_for_Incident_Detection*

Stephanedes, Y. J., Chassiakos, A. P. & Michalopoulos, P. G. (1992), 'COMPARATIVE PERFORMANCE EVALUATION OF INCIDENT DETECTION ALGORITHMS', *Transportation Research Record* (1360).
URL: *http://trid.trb.org/view.aspx?id=370955*

Stephanedes, Y. J. & Hourdakis, J. (1996), 'Transferability of freeway incident detection algorithms', *Transportation Research Record: Journal of the Transportation Research Board* **1554**(1), 184–195.

Stratford, K. & Cowling, A. (2016), Chinese Household Income, Consumption and Savings, Technical report, Reserve Bank of Asutralia.
URL: *https://www.rba.gov.au/publications/bulletin/2016/sep/4.html*

Subramaniam, S. (1991), 'Literature review of incident detection algorithms to initiative diversion strategies', *University Center of Transportation Research, Virginia Polytechnic Institute and State University, Blacksburg, VA, Tech. Rep* .

Sussman, J. S. (2008), *Perspectives on intelligent transportation systems (ITS)*, Springer Science & Business Media.

Suzuki, K. & Jansson, H. (2003), 'An analysis of driver's steering behaviour during auditory or haptic warnings for the designing of lane departure warning system', *JSAE review* **24**(1), 65–70.

Suzuki, S. & Be, K. (1985), 'Topological structural analysis of digitized binary images by border following', *Computer Vision, Graphics, and Image Processing* **30**(1), 32–46.
URL: *http://linkinghub.elsevier.com/retrieve/pii/0734189X85900167*

Tan, H., Xuan, X., Wu, Y., Zhong, Z. & Ran, B. (2016), A Comparison of Traffic Flow Prediction Methods Based on DBN, *in* 'CICTP 2016', American Society of Civil Engineers, Reston, VA, pp. 273–283.
URL: *http://ascelibrary.org/doi/10.1061/9780784479896.026*

Taylor, M. (2019), 'htm-community/htm-school-viz: Visualizations supporting HTM School'.
URL: *https://github.com/htm-community/htm-school-viz*

Taylor, M., Breznak, Surpur, C., Purdy, S., Marshall, A., Ragazzi, D., Ahmad, S., Numenta-ci, Akhila, Weinberger, P., Crowder, R., Simons, C., Vitaly-krugl, McCall, R. J., Lavin, A., Eric, M., Keithcom, Song, U., Yuwei, Borgne, M. L., Bolliger, S., Bridgewater, J., Danforth, I., Weiss, J., Tomsilver, Ray, D., Kamlani, A., Erasmo, H., Gilsho & Blas, E. (2016), 'nupic: 0.5.0'.
URL: *https://doi.org/10.5281/zenodo.46074*

Terry, T. N. & Tanner, S. (2018), 'Problematic Roadway Environments for Automated Vehicles'.

Thogulava, H., Antonovs, V., Bingham, S., Hill, M. & Hanchett, K. (2015), Developing Origin-destination Matrices Using Bluetooth Data for Strategic Transport Models- Worcester Case Study, *in* 'European Transport Conference 2015'.
URL: *https://trid.trb.org/view.aspx?id=1371977*

Thomas, N. (1998), 'Multi-state and multi-sensor incident detection systems for arterial streets', *Transportation Research Part C: Emerging Technologies*

6(5-6), 337–357.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X99000030*

Tian, X., Zhang, J., Ma, Z., He, Y., Wei, J., Wu, P., Situ, W., Li, S. & Zhang, Y. (2017), 'Deep LSTM for Large Vocabulary Continuous Speech Recognition'.
URL: *http://arxiv.org/abs/1703.07090*

Tian, Y. & Pan, L. (2015), Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network, *in* '2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)', IEEE, pp. 153–158.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7463717*

Toroyan, T. (2015), Global Status Report on Road Safety 2015, Technical report, World Health Organisation.

Toroyan, T. (2018), Global Status Report on Road Safety 2018, Technical report, World Health Orgnanisation.

United Nations (2017), World Population Prospects The 2017 Revision, Technical report, United Nations.

Viswanathan, M., Lee, S. H. & Yang, Y. K. (2006), Neuro-fuzzy Learning for Automated Incident Detection, *in* M. Ali & R. Dapoigny, eds, 'Advances in Applied Artificial Intelligence SE - 95', Vol. 4031 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 889–897.
URL: *http://dx.doi.org/10.1007/11779568_95*

Vlahogianni, E. I., Karlaftis, M. G. & Golias, J. C. (2005), 'Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach', *Transportation Research Part C: Emerging Technologies* **13**(3), 211–234.
URL: *https://www.sciencedirect.com/science/article/pii/S0968090X05000276*

Vogiatzis, N., Zito, R. & Stazic, B. (2009), Initial DTEI-SCATS Signalling System Log Reports, Technical report, University of South Australia Institute for Sustainable Systems and Technologies – Transport Systems, Adelaide.

Weil, R., Wootton, J. & García-Ortiz, a. (1998), 'Traffic incident detection: Sensors and algorithms', *Mathematical and Computer Modelling* **27**(9-11), 257–291.
URL: *http://www.sciencedirect.com/science/article/pii/S0895717798000648*

Wibisono, A., Jatmiko, W., Wisesa, H. A., Hardjono, B. & Mursanto, P. (2016), 'Traffic big data prediction and visualization using Fast Incremental Model Trees-Drift Detection (FIMT-DD)', *Knowledge-Based Systems* **93**, 33–46.

Williams, A. H., O'Leary, T. & Marder, E. (2013), 'Homeostatic Regulation of Neuronal Excitability', **8**(1), 1656.

Xie, F. & Levinson, D. (2007), 'Measuring the Structure of Road Networks', *Geographical Analysis* **39**(3), 336–356.
URL: *https://doi.org/10.1111/j.1538-4632.2007.00707.x*

Yan, X. & Han, J. (2003), 'CloseGraph', *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03* **6**, 286.
URL: *https://dl.acm.org/doi/10.1145/956750.956784*

Yuan, F. & Cheu, R. L. (2003), 'Incident detection using support vector machines', *Transportation Research Part C: Emerging Technologies* **11**(3-4), 309–328.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X03000202*

Zhang, C., Zhu, L., Ni, J., Huang, C. & Shen, X. (2020), 'Verifiable and privacy-preserving traffic flow statistics for advanced traffic management systems', *IEEE Transactions on Vehicular Technology* pp. 1–1.

Zhang, K. (2005), UNIVERSAL INCIDENT DETECTION : A BAYESIAN NETWORK APPROACH, PhD thesis, University of South Australia.

Zhang, K. & Taylor, M. A. (2006), 'Effective arterial road incident detection: A Bayesian network based algorithm', *Transportation Research Part C: Emerging Technologies* **14**(6), 403–417.
URL: *http://www.sciencedirect.com/science/article/pii/S0968090X0600088X*

Zhang, K., Vogiatzis, N. & Taylor, M. A. P. (2007), A new design for an intelligent event-responsive urban traffic management system, *in* '30th Australasian Transport Research Forum', Victorian Department of Infrastructure and Bureau of Transprot and Regional Economics.

Zhang, M. (2016), 'Real-time Traffic Flow Prediction using Augmented Reality', *Electronic Theses and Dissertations* .
URL: *http://scholar.uwindsor.ca/etd/5687*

Zhang, T., Ramakrishnan, R. & Livny, M. (1996), 'BIRCH: An Efficient Data Clustering Method for Very Large Databases', *SIGMOD Rec.* **25**(2), 103–114.
URL: *http://doi.acm.org/10.1145/235968.233324*

Zhang, Y., Meratnia, N. & Havinga, P. (2010), 'Outlier Detection Techniques for Wireless Sensor Networks: A Survey', *IEEE Communications Surveys & Tutorials* **12**(2), 1–12.

Zhang, Z. (2018), Artificial Neural Network, *in* Z. Zhang, ed., 'Multivariate Time Series Analysis in Climate and Environmental Research', Springer International Publishing, Cham, pp. 1–35.
URL: *http://link.springer.com/10.1007/978-3-319-67340-0_1*

Zheng, S. (2018), 'China now has over 300 million vehicles'.
URL: *https://www.scmp.com/news/china/economy/article/2088876/chinas-more-300-million-vehicles-drive-pollution-congestion*

Zhou, Y., Chen, S., Mo, Z. & Yin, Y. (2013), Privacy preserving origin-destination flow measurement in vehicular cyber-physical systems, *in* '2013 IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)', IEEE, pp. 32–37.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6614243*

Zoph, B. & Le, Q. V. (2016), 'Neural Architecture Search with Reinforcement Learning'.
URL: *http://arxiv.org/abs/1611.01578*