Enabling Auditing and Intrusion Detection for Proprietary Controller Area Networks

DISSERTATION

Brent J. Stone, Capt, USA

AFIT-ENG-DS-18-D-003

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-18-D-003

ENABLING AUDITING AND INTRUSION DETECTION FOR PROPRIETARY CONTROLLER AREA NETWORKS

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Computer Science

Brent J. Stone, B.S.C.S., M.S.I.T.

Capt, USA

21 November 2018

AFIT-ENG-DS-18-D-003

ENABLING AUDITING AND INTRUSION DETECTION FOR PROPRIETARY CONTROLLER AREA NETWORKS

Brent J. Stone, B.S.C.S.,M.S.I.T.
Capt, USA

Committee Membership:

Scott R. Graham, PhD
Chairman

Barry E. Mullins, PhD
Member

Christine M. Schubert Kabban, PhD
Member

Adedeji B. Badiru, PhD
Dean, Graduate School of Engineering and Management

Abstract

Several trends in the Cyber domain are converging to present an imminent threat to the safety and prosperity of people and property. Efforts by organizations and nations to automate the administration and functions of Cyber-Physical Systems (CPS) such as road vehicles and manufacturing plants have the side effect of connecting previously isolated CPS networks to the global Internet. These CPS networks routinely have ineffective or no Cyber-security measures in place since they were assumed to be isolated from remote access when designed. Cyber attacks have consequently increased in sophistication and scope to exploit these newly connected networks. The threat these trends pose are routinely emphasized by publications and demonstrations by academic, commercial, and government researchers. One particularly troubling area of the growing Cyber threat landscape is the remote exploitation of Internet accessible passenger vehicles.

Several passenger vehicles already being sold in the United States in 2018 have been shown to be accessible from the Internet and vulnerable to Cyber attacks that may allow remote control of the vehicle against the driver's will. Nearly every major automotive manufacturer has subsequently announced plans to incorporate autonomous driving features in their future vehicles. These autonomous driving features are expected to greatly increase the population of Internet accessible vehicles, the complexity of preventing Cyber attacks, and the scope of control an attacker might have should they gain remote control of the vehicle.

A significant increase in Cyber-security research and development is necessary to mitigate the mounting threats to CPS networks in general and automotive networks in particular. Unfortunately, manufacturers often make significant changes to the

proprietary protocols and functionality their CPS rely upon. The CPS manufactures also have several incentives to maintain secrecy regarding these protocols, functions, and changes over time. These incentives include potential advantages competitors may gain from public disclosure, adversity to litigation or loss of customer confidence caused by published findings that products are vulnerable to Cyber attack, and more. The rate of changes to proprietary protocols, ubiquitous secrecy by manufacturers, expense and difficulty of attaining working CPS, and other factors all inhibit the scope and scale of independent Cyber-security research in the CPS network domain. Automating some of the time consuming and difficult analysis commonly associated with CPS Cyber-security research may help mitigate these challenges.

This dissertation presents a series of techniques intended to automate the analysis of proprietary non-text network protocol payloads used by Cyber-Physical Systems (CPS). These techniques include unsupervised *lexical analysis* methods for extracting logically distinct pieces of information from non-text payloads and then enumerating the correlated and causal relationships that exist among that information. Analyzing the Controller Area Network (CAN) protocol used by CPS in passenger vehicles, medical electronics, and industrial Internet of Things is used as a proof of concept validate the consistency and accuracy of these techniques. Empirical validation is performed using data collected from seventeen passenger vehicle CAN networks. This research also proposes a series of validation strategies which do not rely upon truth data. These validations techniques are intended to enable quantifiable research findings regardless of whether access to proprietary truth data is available.

This research represents the first robust collection of techniques and validation strategies for automating analysis of CPS network protocols. The avoidance of heuristics and a focus on techniques from the Machine Learning and Life Sciences domains help ensure this research is generalizable to multiple types of CPS and non-text net-

work protocols. Furthermore, all of the proposed techniques are unsupervised meaning that no *a priori* information or truth data is needed to effectively use them.

These unsupervised methods serve as solutions for two problems related to improving the Cyber-security posture of CPS networks. First, they improve the throughput and scope of independent Cyber-security audits of proprietary networks by automating the discovery of data and their relationships. Second, the collective output of these techniques forms an expansive feature set for the development and testing of automated auditing methods and Intrusion Detection Systems. Findings generated using these methods are also expected to accelerate the availability of labeled data sets which might further improve the frequency, accuracy, and scope of published Cyber-security research for the CPS network domain.

Acknowledgements

My deepest thanks to my committee, mentors, peers, and family for their guidance, inspiration, and support. My thinking formed this dissertation; they formed my thinking.

Brent J. Stone

Table of Contents

Page

Page

List of Figures

List of Tables

List of Abbreviations

ENABLING AUDITING AND INTRUSION DETECTION FOR PROPRIETARY CONTROLLER AREA NETWORKS

I. Introduction

1.1 Motivation

New methods are consistently developed to remotely gain unauthorized access to computing systems for fun and profit. While hardware and software vendors usually develop and deploy patches or fixes to stop these exploits, independent cyber-security research has unquestionably been an important driver in improving the pace and quality of those counter-measures. The impending proliferation of cyber-physical systems (CPS) incorporating persistent Internet connections such as self-driving passenger vehicles and medical electronics presents a growing cyber-security challenge. Unfortunately, the secretive and proprietary nature of the automotive industry in particular makes documenting vehicle networks and their CPS a time consuming task. Such documentation is necessary for independent researchers to perform security audits or develop, test, and validate cyber-security defenses like Intrusion Detection Systems (IDS).

Security audits of in-production Internet accessible vehicles by Miller and Valasek, Koscher et. al., and Nie et. al. have clearly demonstrated that independent cyber-security analysis of automotive networks and CPS are necessary to ensure people, property, and private information are protected from cyber attacks [21, 22, 23]. In those examples, as with security audits of traditional computing environments, reconnaissance and enumeration of the network and systems is a necessary precursor

to identifying flaws that could be maliciously exploited.

The 2015 Volkswagen emission scandal also highlights the possibility that automotive manufacturers have competitive and financial incentives to deliberately introduce and conceal misconfigured CPS in their vehicles [24]. At least seven other major Original Equipment Manufacturers (OEMs) have since been found to use similar tactics [25, 26]. These findings further emphasize the importance and urgency for methods to automate and improve independent cyber-security research in the automotive sector. They also highlight the need for similar work in other industries, such as medical electronics and robotics, which may have similar business incentives and technology [27].

This dissertation focuses on automotive CPS using the Controller Area Network (CAN) protocol to frame the discussion and findings. This particular domain is used for four reasons. First, the current scale of self-driving vehicle research and development taking place in 2018 presents a significant cyber-security challenge in the near future. Second, the CAN protocol is used by a wide range of industries. Third, there is a standardized and expedient method for collecting CAN network samples from passenger vehicles. Fourth, it is relatively easy to find research volunteers who own a passenger vehicle.

1.2 Research Questions and Document Outline

The following research questions are intended to support the overall objective of improving the pace and quality of independent research and development of robust cyber-security measures for networks using CAN and similar protocols. These questions represent a decomposition of that overall objective into two fundamental tasks: identify the logically distinct data present in a proprietary CPS network and then discover relationships between those data. The first task is referred to as *lexical analysis*

and the second as *semantic analysis*. The concepts of *lexical analysis* and *semantic analysis* are discussed in detail in Sections 2.4 and 2.5.

By developing *unsupervised* methods for accomplishing these tasks, independent researchers can quickly gain an initial understanding of a particular CPS network necessary to pursue their specific goals. The term *unsupervised* refers to methods which require little or no input beyond the data set being analyzed. The need for a validation strategy for each proposed *unsupervised* method is the basis for the third research question.

- **Research Question 1:** Is there a robust method for *unsupervised lexical analysis* of proprietary automotive CAN payloads given little or no truth data?
- **Research Question 2:** What *unsupervised semantic analysis* of automotive *time series* is possible given little or no truth data?
- **Research Question 3:** What are viable validation methods for *unsupervised lexical and semantic analysis* of automotive CAN data when little or no truth data are available?

Chapter II provides information about cyber-physical systems (CPS), the CAN protocol, *lexical* and *semantic analysis*, automated reverse engineering, Empirical Data Modeling (EDM), and other related work. Data collection and related research methods are discussed in Chapter III. Research Question 1 is the focus of Chapter IV. Chapters V and VI present *semantic analysis* techniques to address Research Question 2. Validation methods related to Research Question 3 are discussed in each chapter. Table 1 provides a summary of the relationship between research questions and chapters as well as how the output of each chapter serves as the input to the following chapter. Final conclusions are presented in Chapter VII followed by a brief discussion of future work in Chapter VIII.

Table 1. Outline of the Input and Output Linking Research Questions and Their Associated Chapters

	Data Collection *Chapter III*	**Research Question 1** Lexical Analysis *Chapter IV*	**Research Question 2** Semantic Analysis *Chapters V & VI*
Input	- Driving Route - Driving Conditions - Driver Input	*Output of Data Collection*	*Output of Lexical Analysis*
Output	- Arbitration IDs - Payloads	- Payload Compositions - Time Series (Signals)	- Correlated Relationships - Causal Relationships - Predictive Models

1.3 Contributions

The major contributions of this dissertation are the following:

1. The first known unsupervised *time series* extraction pipeline for the CAN protocol used by CPS networks in the automotive, medical, and robotics industries, among others.

2. The combination of Empirical Dynamic Modeling (EDM) with the *time series* extraction pipeline to enumerate correlated and causal relationships in vehicular networks and create accurate models of those networks without truth data.

3. A comprehensive set of metrics and strategies for evaluating unsupervised *lexical* and *semantic analysis* methods for CAN networks when little or no truth data is available.

II. Related Work

Each section in this chapter focuses on a particular field of research or technology. A brief background on cyber-physical systems (CPS) and *time series* data is provided in Section 2.2[1]. Section 2.3 introduces the Controller Area Network (CAN) protocol and its use in the automotive industry. The concept of *lexical analysis* in the context of CAN payload reverse engineering (RE) is introduced in Section 2.4. Previous network protocol RE proposals are discussed in Section 2.5. Section 2.5 also introduces the RE pipeline linking Chapters IV and V and its influences from prior work. The unsupervised clustering method used in that pipeline—Agglomerative Hierarchical Clustering—is introduced in Section 2.6. Section 2.7 discusses Empirical Data Modeling (EDM) techniques used in Chapter VI followed by an introduction to verification and validation (V&V) techniques in Section 2.8.

The dissertation uses the field of compiler design to frame the terminology, techniques, and objectives of Chapters IV and V. This decision is based on the reality that there is a *definitively correct interpretation* for every payload in a CAN network and data identified using that *interpretation* represents some specific logical or physical process. That *definitive* interpretation was engineered by the vehicle's Original Equipment Manufacturer (OEM) and is confidentially shared with mechanics and component manufacturers via a Database CAN (DBC) file. This closely mirrors the problem of designing a computer program compiler which must interpret a program according to the definitive rules specified by the programing language [29].

Proposed automated RE approaches in a domain which feature *deterministic* relationships between input and its interpretation should therefore resemble compiler design. The field of natural language processing (NLP) is a testament to this observation. In almost every sub-field of study in NLP, from machine-aided translation

[1] A *time series* is a univariate sequence of values ordered by the time observed [28].

to data mining, the first tasks performed are *lexical* and *semantic* analysis of the raw data [30, 31, 32, 33, 34, 35]. Thus, the RE pipeline linking Chapters IV and V is intended to decompose the task of automated RE using the proven sequence of subtasks used by compiler design and NLP. This framework of loosely coupled *lexical* and *semantic analysis* subtasks frames the sequence and scope of Chapters IV and V and their associated research questions proposed in Section 1.2. However, how unsupervised *lexical* and *semantic analysis* can be implemented without truth data or significant *a priori* information remains a largely unexplored area in published research.

A wide range of fields were considered in pursuit of robust techniques, metrics, and features which are viable for unsupervised *lexical* and *semantic analysis* in the context of CAN payload reverse engineering (RE). Examples of this breadth include consideration of the Needleman-Wunsch and BLAST sequence alignment algorithms from the bioinformatics community, various statistical machine learning methods, linear regression techniques, and slightly more exotic mathematical methods such as wavelet ANOVA, exploratory factor analysis (EFA), and principal component analysis (PCA) [28, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46]. Each of these techniques have been used to answer research questions similar to those offered in Section 1.2; however, none of them were found to be viable or sufficiently robust in the context of automotive network reverse engineering (RE) and analysis. Ultimately, the approaches presented in Chapters IV, V, and VI address the research questions using a combination of nine concepts employed as summarized in Table 2.

2.1 Reverse Engineering and Expert Systems

The use of greedy search and a bespoke validation metric—*alignment score*—in Chapters IV and V may justifiably raise concern that the proposals are some kind of

Table 2. Summary of Lexical Analysis, Semantic Analysis, and Validation Techniques Discussed

	Data Collection *Chapter III*	Payload Lexical Analysis *Chapter IV*	Time Series Semantic Analysis *Chapters V & VI*
Implementation		- Exclusive Or - Greedy Search	- Shannon Diversity Index - Pearson's ρ - Agglomerative Clustering - EDM[a]
Validation	- Jaccard Index - Dynamic Time Warping	- Jaccard Index - Alignment Score[b]	- Jaccard Index - EDM

[a]EDM encompasses several techniques based upon Takens' Theorem. See Section 2.7
[b]Alignment Score is novel metric introduced in Section 2.8

expert system rife with unadvertised errors, faulty assumptions, or simply too targeted to be of serious academic or commercial interest. The RE pipeline introduced in Section 2.5 which links Chapters IV and V meets one definition of an *expert system*: "a computer model of expert human reasoning, reaching the same conclusions the expert would reach if faced with a comparable problem" [47]. One type of *expert system* by this definition are "rule-based systems based upon identifiable human expertise [...such as...] Exper-TAX [...] to give advice on corporate tax planning, ONCOCIN [...] which helps doctors determine appropriate treatments for chemotherapy patients, and CLASS [...which] supports commercial loan decisions in a bank" [47].

As of 2018 *expert system* research has been dwarfed by statistical machine learning, deep learning, and statistical analysis methods. Thus, proposals resembling an *expert system* understandably raise concern that they are ultimately a failure to understand and apply these newer methods. The automated RE proposals discussed in Section 2.5 often use some of these newer analysis techniques; however, they may still fit the definition of *expert systems* due to heavy reliance on heuristics.

The automated RE pipeline detailed in Chapters IV and V differs from 20^{th} cen-

tury *expert systems* and many other automated RE proposals in that there is no use of heuristics. Instead, the reverse engineering (RE) effort is reduced to a series of distance measurements and tunable threshold parameters which is functionally similar to statistical machine learning methods. This is not to say there is no expert knowledge incorporated into these methods. However, the insights focus on system or data encoding idiosyncrasies rather than heuristic assumptions about data produced by the system.

The approach in Chapter IV depends upon insight that the *least significant bit* (LSB) of continuous numerical data transitions more frequently than the *most significant bit* (MSB). The techniques presented in Chapter V rely upon the insight that continuous numerical data in automotive networks reliably have a larger Shannon Index—a measure of population diversity—than non-continuous data [48]. Chapter V also relies on the observation that automotive networks necessarily contain sets of highly correlated *time series*. An example of this observation is that passenger vehicles always have two or more wheels which exhibit highly correlated behavior such as rotations per minute (RPM) over time.

Research in the field of automated RE is a delicate balance between reducing and defining a problem space using expert human insight while developing methods which are robust by not relying on that same (potentially faulty, biased, incomplete, or inaccurate) insight. We propose that the automated RE methods presented in Chapters IV and V strike an appropriate balance between the competing goals of using expert insight to identify and understand the problem space while remaining robust with respect to variations in network configuration and operational conditions. This position is supported by the overwhelmingly positive findings presented in Sections 4.4 and 5.4.

Relying on Life Sciences Research.

The automated reverse engineering (RE) proposals in Chapters IV and V have very little methodological similarity with the superficially similar proposals discussed in Section 2.5 for three important reasons. The first is the opinion that many of those proposals are effectively *expert systems* due to heavy reliance on *expertly developed* heuristics. The RE methods presented in Chapters IV and V deliberately avoid heuristics. The second differentiating factor is practically all published research is focused on analyzing text-based application layer protocols [50, 49, 9, 51, 52]. The methods presented in this paper are based on analyzing payloads of a single known protocol-Controller Area Network (CAN)-which is not text-based. The key differences are the difficulty of *lexical analysis* and the assumption that the vast majority of data are *time series* generated by a CPS. Third, many of these proposals attempt to perform *lexical* and *semantic* analysis simultaneously. Research in the fields of compiler design and NLP demonstrate that these tasks should be independent.

We propose the difference between translating sentences written using Japanese Katakana and English is a reasonable analogy to the difference in *lexical analysis* between existing research and the problem addressed by this research. Automated translation of either language certainly shares similarities once the words in a sentence and their ordering have been identified. However, with text-based network protocols and English there is a finite set of delimiters that are almost always present between 'words'. Thus, the lexical analysis phase proposed in related work is almost always a trivial process using a set of delimiter characters known *a priori* [50, 49, 9, 51, 52]. Sentences written with Japanese Katakana and CAN payloads do not use explicit delimiters. This makes 'word' discovery non-trivial in these contexts.

Another distinguishing factor between the research presented in this dissertation and other automated network protocol RE proposals is the scale of potential *types*

of data present in the network. Much of the discussion in proposals for reverse engineering (RE) text-based protocols deals with differentiating different *types* of data such as text, numbers, and various meta-data. Automotive networks are assumed to be almost exclusively populated by cyber-physical systems (CPS) which produce *time series* data. This difference has a significant impact on the assumptions and techniques that are viable in either context.

Avoidance of heuristics, the difficulty of *lexical analysis*, and the predictable presence of *time series* data are the reasons this chapter includes very little analysis about how to build upon, improve, or otherwise be significantly informed by other automated RE research. These factors also explain why this dissertation instead draws heavily from life sciences research. Techniques from the life sciences domains—such as Empirical Data Modeling (EDM)—are frequently developed with the expectation that no definitive truth data exists to test research hypothesis or develop models [53]. Those techniques are also commonly focused on examining *time series* data. Thus, the statistics and techniques used throughout this dissertation are predominantly re-purposed from research in the life science domains.

2.2 Cyber-Physical Systems and Time Series Data

The Association for Computing Machinery(ACM) Transactions on Cyber-Physical Systems (TCPS) defines Cyber Physical Systems (CPS) as follows:

> Cyber-Physical Systems (CPS) has emerged as a unifying name for systems where the cyber parts, i.e., the computing and communication parts, and the physical parts are tightly integrated, both at the design time and during operation. Such systems use computations and communication deeply embedded in and interacting with physical processes to add new capabilities to physical systems. These cyber-physical systems range from miniscule (pace makers) to large-scale (a national power-grid). [54]

The output of CPS monitoring physical processes is often *time series* data representing the state of those processes over time. Examples of *time series* in a passenger vehicle might be measurements by an Electronic Control Unit (ECU) monitoring the front right wheel's rotations per minute (RPM), steering wheel angle, or engine RPM. Analysis of *time series* data is an active field of study with many unique and interesting approached. The National Institute of Science and Technology (NIST) frames the field of *time series* analysis in the following terms:

> Time series analysis accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. [28]

A requirement for many univariate *time series* analysis techniques is that the data are *stationary*. "A stationary process has the property that the mean, variance and autocorrelation structure do not change over time. Stationarity can be defined in precise mathematical terms, but for our purpose we mean a flat looking series, without trend, constant variance over time, a constant autocorrelation structure over time and no periodic fluctuations (seasonality)" [28]. Fig. 1 presents some examples of *time series* with plots (b) and (g) representing examples of *stationary time series*.

Time series analysis techniques requiring *time series* are *stationary* are not viable for *time series* produced by automotive CPS networks. The driver input and driving conditions are arbitrary and produce data which do not fit the definition of *stationary time series*. Exponential smoothing and Auto-Regressive Integrated Moving Average (ARIMA) models are examples of *time series* analysis methods intended to enable modeling *non-stationary time series* [1, 44]. EDM techniques introduced in Section 2.7 are compelling methods for analyzing *non-stationary time series* and the focus of Chapter VI.

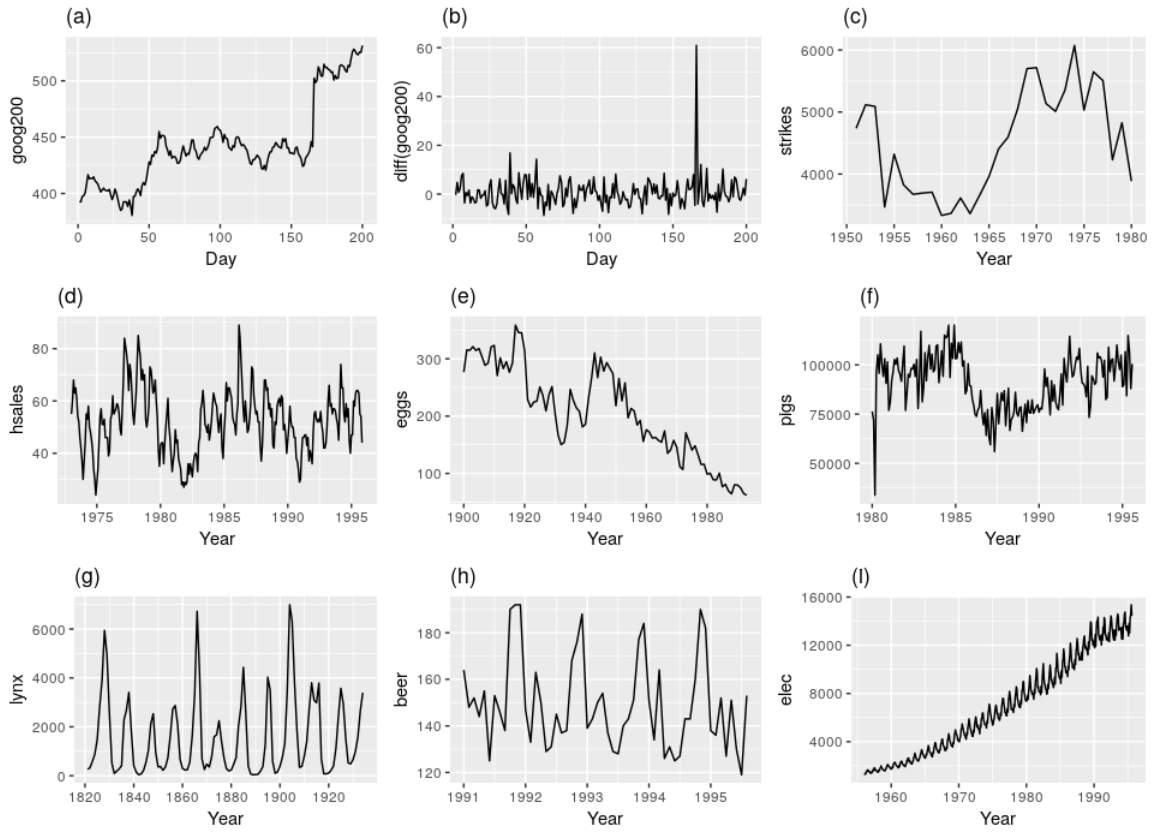


Figure 1. Examples of Time Series Plots: "(a) Google stock price for 200 consecutive days; (b) Daily change in the Google stock price for 200 consecutive days; (c) Annual number of strikes in the US; (d) Monthly sales of new one-family houses sold in the US; (e) Annual price of a dozen eggs in the US (constant dollars); (f) Monthly total of pigs slaughtered in Victoria, Australia; (g) Annual total of lynx trapped in the McKenzie River district of north-west Canada; (h) Monthly Australian beer production; (i) Monthly Australian electricity production." [1]

2.3 Controller Area Network (CAN)

The CAN protocol (ISO 11898) and its physical bus design were developed and patented by a German company named Bosch in the 1980s [55, 56]. It is a data-link layer protocol which is commonly used in the automotive, industrial manufacturing, and medical electronic industries, among others, due to its relatively low cost, low physical weight, and architectural simplicity [27, 55, 57, 58, 59, 60]. A CPS network using the CAN protocol is typically implemented using one or more frame formats.

These formats include the standard format shown in Fig. 2, the extended format shown in Fig. 3, and the Flexible Data Rate (CAN-FD) format shown in Fig. 4. These formats are backwards compatible. The extended format enables more message IDs. The CAN-FD format enables payload sizes larger than eight bytes and a potential transmission rate increase from 1Mbps to 4Mbps [3, 61]. Only the arbitration ID (Arb ID) and payload sections of the frame are variable in all three message formats. The remaining portions of the frame are flags and error checking bits with have a deterministic relationship to the Arb ID and payload used.

CAN in the Automotive Industry.

Automotive CAN networks are used as the proof of concept for analyzing proprietary CPS networks based on stateless binary protocols for three reasons. First, U.S. federal regulation requires passenger vehicles produced in 2008 and later to provide an On Board Diagnostics (OBD)-II interface in the cabin which uses the CAN protocol for communication [62, 63, 64, 65]. This facilitates relatively easy collection of real network data. Second, we empirically found that most passenger vehicles use the standard or extended frame formats specified in ISO 11898-1:2015 [2]. This means the maximum data per CAN frame is eight bytes. This bounds the complexity of examining CAN network traffic logs. Third, the requirement to use CAN in all passenger vehicles ensures a standardized and relatively easy method is available to collect logs from a range of manufacturers, vehicle types, and model years. Testing reverse engineering and analysis techniques using a breadth of network configurations is intended to improve confidence that those techniques are robust.

The multi-cast and clear-text nature of CAN bus networks are the primary reasons data collected using a vehicle's OBD-II interface is effective. Unless the manufacturer specifically isolates the interface using some kind of filter, many of the CAN frames

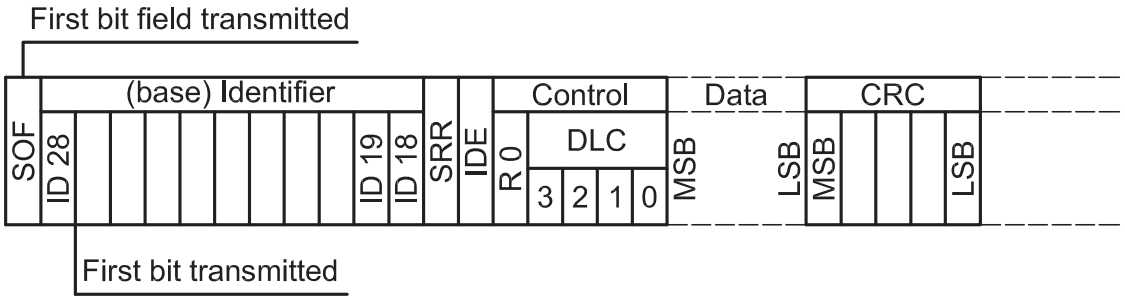


Figure 2. ISO 11898-1 Controller Area Network Standard Message Format [2]

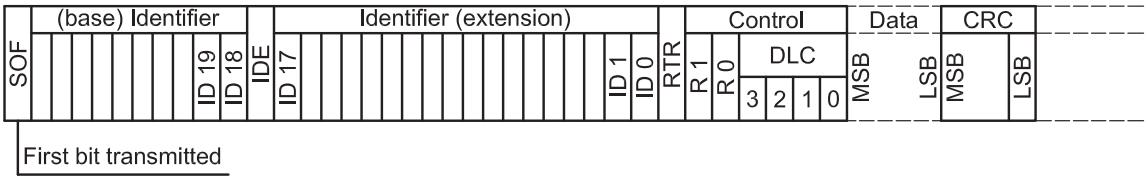


Figure 3. ISO 11898-1 Controller Area Network Extended Message Format [2]

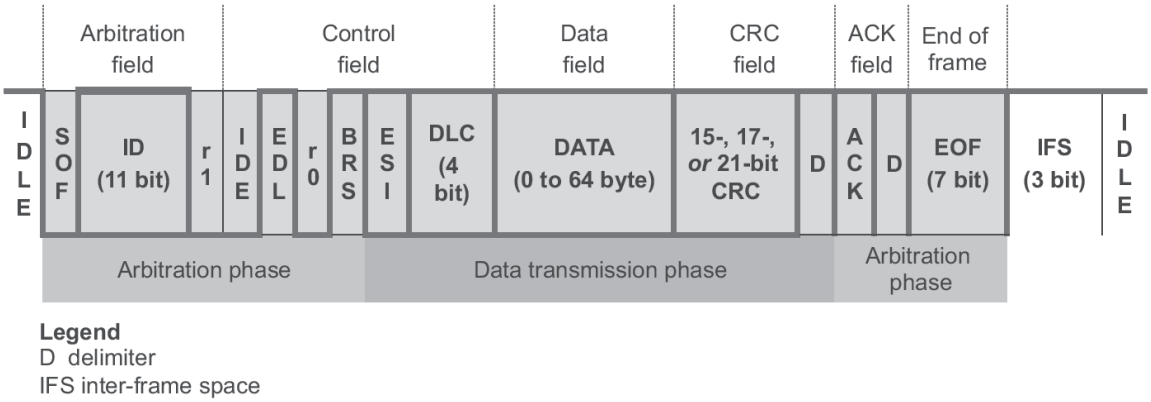


Figure 4. ISO 11898-1 Controller Area Network Flexible Data Rate Message Format [3]

normally transmitted by Electronic Control Units (ECU)[2] in the vehicle are observable from the OBD-II interface. Only one vehicle studied appeared to implement filtering in this manner[3]. Fig. 5 is an example wiring harness used by a passenger vehicle. One end point of the harness will be the OBD-II interface which is located under the steering column in passenger vehicles.

Table 3 provides a summary of the proprietary and non-proprietary standards

[2]Electronic Control Unit (ECU) is the term commonly used to refer to cyber-physical systems in the automotive industry

[3]The 17 vehicles studied all included an internal combustion engine. Fully electric vehicles studied appear to use Automotive Ethernet instead of CAN as their primary network protocol [4, 23, 66]. These electric vehicles did not generate CAN data observable from the OBD-II interface during normal use

Figure 5. A Typical Wiring Harness In A Passenger Vehicle [4]

Table 3. Legislated emissions-related OBD/WWH-OBD diagnostic specifications applicable to the OSI layers [20]

<table>
<tr><th>Applicability</th><th>OSI 7 layers</th><th colspan="6">Emissions-related OBD communication requirements</th><th colspan="3">Emissions-related WWH-OBD communication requirements</th></tr>
<tr><td rowspan="9">Seven layer according to ISO/IEC 7498-1 and ISO/IEC 10731</td><td>Application (layer 7)</td><td colspan="6">ISO 15031-5/SAE J1979</td><td colspan="3">ISO 27145-3</td></tr>
<tr><td rowspan="2">Presentation (layer 6)</td><td colspan="6">ISO 15031-2, ISO 15031-5, ISO 15031-6</td><td colspan="3">ISO 27145-2</td></tr>
<tr><td colspan="6">SAE J1930-DA, SAE J1979-DA, SAE J2012-DA</td><td colspan="3">SAE J1930-DA, SAE J1979-DA, SAE J2012-DA</td></tr>
<tr><td>Session (layer 5)</td><td colspan="6">Not Applicable</td><td colspan="3">ISO 14229-2</td></tr>
<tr><td>Transport (layer 4)</td><td colspan="2" rowspan="2">ISO 15031-5</td><td>ISO 14230-4</td><td>ISO 15765-2</td><td rowspan="3">ISO 15765-4</td><td>ISO 15765-2</td><td rowspan="3">ISO 15765-4</td><td>ISO 13400-2</td></tr>
<tr><td>Network (layer 3)</td><td colspan="3"></td><td></td><td></td><td></td></tr>
<tr><td>Data link (layer 2)</td><td rowspan="2">SAE J1850</td><td rowspan="2">ISO 9141-2</td><td>ISO 14230-2</td><td rowspan="2">ISO 11898-1, -2</td><td></td><td rowspan="2">ISO 11898-1, -2</td><td></td></tr>
<tr><td>Physical (layer 1)</td><td>ISO 14230-1</td><td></td><td>ISO 13400-3</td></tr>
</table>

used by light duty and heavy duty (>8,500 pounds) vehicles in the United States. Note that most of these standards continue to use CAN as the data-link layer for their implementation. While these standards are published by the Society of Automotive Engineers (SAE) and the International Standards Organization (ISO), CAN frame payloads in passenger vehicles primarily use unpublished proprietary formatting which varies by manufacturer, model, and model year [2, 67, 68].

Accurate payload interpretation is only possible using Data Base CAN (DBC) files provided by the Original Equipment Manufacturer (OEM) to licensed mechanics and equipment vendors. At least one manufacturer even prevents mechanics from directly accessing this information by requiring all vehicle network Measurement, Calibration, and Diagnostics (MCD) services be done over the Internet using remote OEM servers [69]. Fig. 6, 7, and 8 are examples of the proprietary hardware and software used by mechanics to access and use DBC files to interpret a vehicle's CAN payloads.

Access to OEM provided DBC files may not guarantee a completely accurate view of a vehicle's network. This reality has become more apparent since the 2015 Volkswagen emission scandal caused increased scrutiny across the industry [24]. As of 2018, at least seven other OEMs have been found to manufacture vehicle networks

Figure 6. wiTECH Micropod II Diagnostics and ECU Reprogramming Tool for Vehicles Manufactuered by Fiat Chrysler Automobiles (FCA). [5]

which appear to deliberately misrepresent at least emissions and power train information gathered using SAE J1979 diagnostics information requested using the OBD-II interface [20, 25, 26]. These kinds of OEM generated faults are especially concerning considering that the CAN protocol in particular and OEM financial incentives in general may also apply to the medical equipment and robotic manufacturing industries [27, 59].

Society of Automotive Engineers (SAE) J1979 Diagnostic Protocol.

SAE J1979 is a useful source of truth data for testing automated reverse engineering methods regardless of potential deliberate misconfiguration by OEMs. SAE J1979 is the United States (U.S.) federally mandated diagnostics standard for requesting information from light duty passenger vehicles [20, 62, 70]. The standard consists of nine different *service* modes with mode \$01 corresponding to requests for the current state of the vehicle. Requests for information are made using a Parameter ID (PID) with each PID in a given *service* mode corresponding to a particular function in the vehicle. Table 4 lists some PIDs for mode \$01 which are useful features in the context of testing automated CAN payload reverse engineering techniques. All ECUs

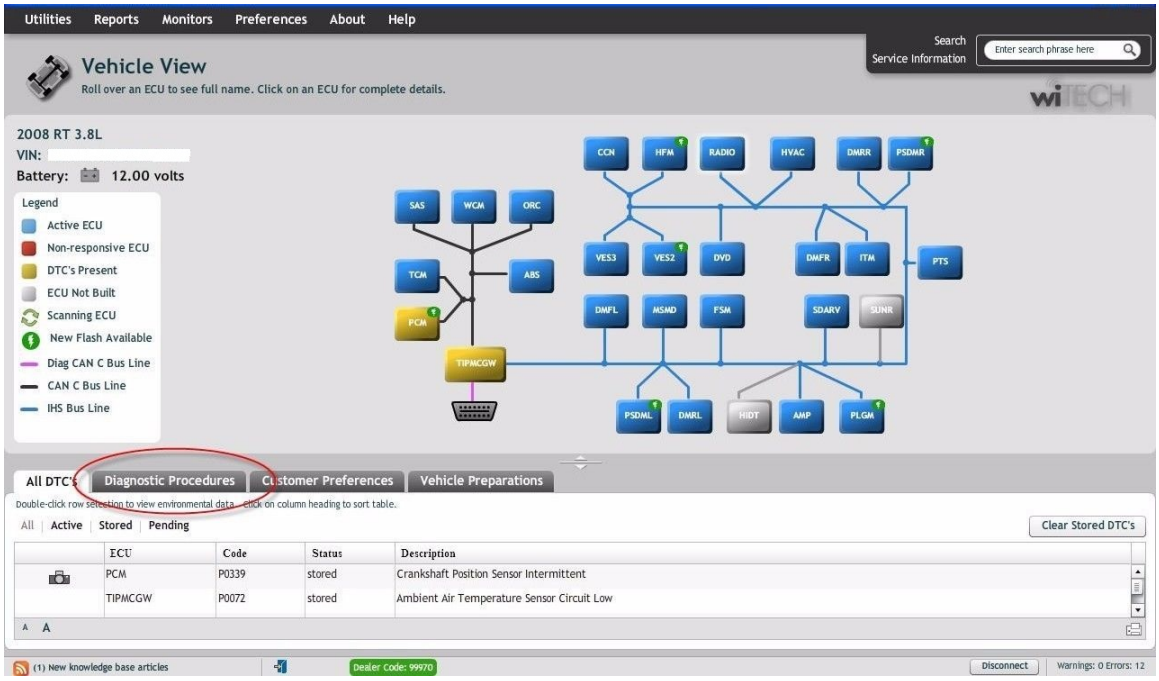


Figure 7. wiTECH Diagnostics and ECU Reprogramming Software for Vehicles Manufactuered by Fiat Chrysler Automobiles (FCA). [6]

Figure 8. Honda / Acura Modular Vehicle Communications Interface (MVCI). [7]

in a vehicle capable of supporting a diagnostic must respond with the specific PIDs they support when a broadcast request is made using *service* $01 PID $00 [20]. This allows researchers to determine which diagnostic information they can request from a particular vehicle during data collection.

Table 4. Examples of SAE J1979 Service $01 Diagnostic Requests Related to Locomotion and Driver Input [20]

PID (hex)	PID (Dec)	Bytes Returned	Description	Minimum Value	Maximum Value	Units	Formula
0C	12	2	Engine RPM	0	16,383.75	rpm	$\frac{256A+B}{4}$
0D	13	1	Vehicle speed	0	255	km/h	A
11	17	1	Throttle position	0	100	%	$\frac{100}{255}A$
45	69	1	Relative throttle position	0	100	%	$\frac{100}{255}A$
5A	90	1	Relative accelerator pedal position	0	100	%	$\frac{100}{255}A$
61	97	1	Driver's demand engine - percent torque	-125	130	%	$A-125$
62	98	1	Actual engine - percent torque	-125	130	%	$A-125$
63	99	2	Engine reference torque	0	65,535	Nm	$256A+B$

2.4 CAN Payload Lexical Analysis

The concept of *lexical analysis* is taken from compiler design in computer science. Compilers are the software tools which convert a program into a series of operations that can run on computer hardware [29]. *Lexical analysis* is the first step of a compiler which uses human readable programming code as an input. The first step of *lexical analysis* is the *tokenization* process which identifies the individual logical units, or *tokens*, that code consists of. For example, the following code snippet results in the nine *tokens* `‘for’, ‘x’, ‘in’, ‘range’, ‘(’, ‘0’, ‘,’, ‘10’, ‘)’`:

```
for x in range(0, 10)
```

If the `f` and `o` in the token `‘for’` are incorrectly separated during *tokenization*, then the subsequent steps in the compiler should fail. The compilation should also

fail if the tokens '(' and '0' are *not* separated during *tokenization*.

We define the *tokenization* of CAN payloads as the process of identifying the logically distinct *time series* present within payloads using the same arbitration ID. Individual *time series* extracted through *tokenization* are referred to as *signals* or *tokens* for the remainder of this dissertation.

As an example, imagine the RPM *signals* for two of a vehicle's wheels and a checksum are all contained in the set of 64-bit payloads using a CAN arbitration ID of 0xA15. The two RPM measurements and checksum are each 8-bit *signals*. A possible *tokenization* would be the set of start and stop indices: (0, 7), (8, 15), (56, 63). The *bit positions* 16 through 55 are *padding bits* which are consistently 1 or 0 in every observed payload using ID 0xA15. Fig. 9 depicts this hypothetical *tokenization* scenario.

We empirically found that *tokenization* is necessary to correctly interpret CAN message payloads. This is because a series of payloads using a shared arbitration ID often contains multiple sensor readings concatenated together. This observation is echoed by other third party automotive CAN research findings [8, 21]. The correct *tokenization* for a particular arbitration IDs (Arb ID) payload is known by vehicle Original Equipment Manufacturers (OEMs) but strictly protected as trade secrets.

The following definitions and names are used when discussing the payload *tokenization* process.

- Let X_{ID} refer to the matrix of chronologically ordered payloads using Arb ID

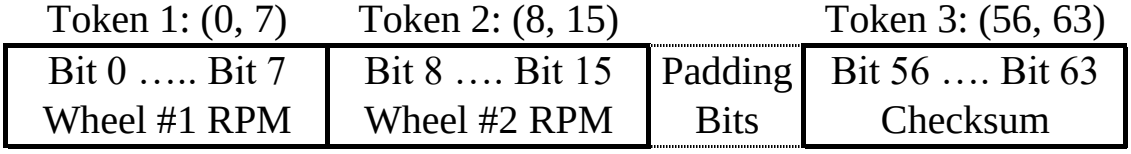


Figure 9. Example Tokenization Result for a 64-bit CAN Payload

'ID'.

  - Let $X_{ID,i}$ refer to the *ith* row in the matrix X_{ID}. This row represents one observed payload of n bits.

- X_{ID} is comprised of an ordered sequence of *bit positions* $x_j \in X_{ID}$ where $0 \leq j \leq n$. n is the bit width of the matrix of payloads X_{ID}.

  - $x_0 \in X_{ID}$ refers to the leftmost *bit position* in the set of payloads X_{ID}. $x_n \in X_{ID}$ refers to the right most *bit position*.

  - This reference format is for notation purposes only and is not intended to imply that the set of payloads X_{ID} use big-endian bit ordering[4].

- For a given sample of observed CAN traffic, let X_{ID} be represented by an $I \times J$ matrix with i observed payloads each comprised of j *bit positions*.

- Let each *time series token* embedded in the set of payloads X_{ID} be represented by t_k. k is the index for a given token within a *composition* $Comp_m(X_{ID})$ of X_{ID}.

  - The term *composition* of n is taken from combinatorics and refers to the set of all ordered sets of positive integers whose sum equals n [71]. For example, the *compositions* of 3 is the set {*(1+1+1), (1+2), (2+1), (3)*}.

  - There are 2^{n-1} unique *compositions* for a payload of bit width n [71].

  - The subscript m in $Comp_m(X_{ID})$ is used to differentiate unique *compositions* of X_{ID}.

[4]The terms *big endian* and *little endian* refer to the decision to place the most significant bit for a piece of data in the first or last memory address. This decision is often expressed as deciding whether the left most bit in a series of bits represents the *most significant bit* or the *least significant bit*. An analogy to written language is whether words should be read from right to left or left to right

Using these definitions, the example shown in Fig. 9 may be represented by the set of tokens $\{t_0, t_1, t_2, t_3\}$. t_0 and t_1 correspond to the two wheel RPM signals, t_2 the padding bits, and t_3 the checksum. t_0 is the set of *bit positions* $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$. The *composition* $Comp_m(X)$ for Fig. 9 is the ordered set of *token* bit widths $\{8, 8, 40, 8\}$. There are $2^{64-1} = 9,223,372,036,854,775,808$ possible *compositions* of this 64-bit payload.

Assumptions and Hypothesis for CAN Payload Tokenization.

The following hypothesis are made regarding CAN payload *lexical analysis*.

- ***Lemma 1:*** $\forall X_{ID,i} \in X_{ID} : ||X_{ID,i}|| = ||X_{ID,i+1}||$.
  - For all observed CAN message payloads $X_{ID,i}$ for a given ID, the cardinality (bit width) of the payloads are constant. Empirical findings in this dissertation and reports by Miller and Valasek have shown this is a reasonable assumption for the vast majority of CAN frames observed in passenger vehicle CAN bus networks [21]. Developing methods to analyze payload populations which violate this lemma is reserved for future work.
- ***Lemma 2:*** $\forall x_j \in X_{ID}\ \exists t_k \in Comp_m(X_{ID}) : x_j \in t_k \wedge x_j \notin t_{n \neq k}$.
  - For every bit position x_j in a set of payloads $X_{ID,i}$, there exists one and only one token t_k in a given *composition* $Comp_m(X_{ID})$ which contains x_j. This lemma ensures that all *bit positions* x_j comprising X_{ID} are accounted for in every $Comp_m(X_{ID})$ and the set of *time series tokens* t_k defining each $Comp_m(X_{ID})$ are disjoint sets of *bit positions* x_j.
- ***Lemma 3:*** $\forall X_{ID,i} \in X_{ID} : Comp_m(X_{ID,i}) \equiv Comp_m(X_{ID,i+1})$
  - The *composition* of all observed payloads $X_{ID,i}$ using arbitration ID 'ID' are identical. As with Lemma 1, this assumption is reasonable based

on empirical research using passenger vehicles manufactured by several OEMs. Developing methods to analyze payload populations which violate this lemma is reserved for future work.

The input of CAN *tokenization* is a series of chronologically-ordered CAN payloads which share the same arbitration ID X_{ID} within a sample of CAN network traffic. The output of CAN *tokenization* is the *composition* $Comp_m(X_{ID})$ of X_{ID} which enables correct interpretation of each logically-distinct *signal* within X_{ID}. *Signals* extracted in this manner also form part of the output of CAN payload *lexical analysis*.

Correct payload *tokenization* and labeling of CAN *signals* enables a broad range of research. Chapters V and VI use the output of CAN payload *tokenization* to search for correlated and causal relationships among *signals* present in a vehicle CAN network. Chapter VI also uses these *signals* to make predictive models of each other. Related research has shown that it is possible to identify who is driving a vehicle out of a population of known drivers using the brake pedal *signal* [40].

2.5 Automated Reverse Engineering (RE) of Network Protocols

The concept of automated protocol reverse engineering (RE) using observed network traffic is an active area of research. Practically all published research is focused on analyzing text-based protocols with the goal of facilitating *deep packet inspection* [9, 49, 50, 51, 52]. Prior research has uncovered multiple remote exploits of Internet capable vehicles which includes being able to take full control of the vehicle [21, 23, 72, 73, 74]. However, these findings were achieved through manual RE and exploit development. These manual RE efforts included RE a limited number of CAN payload *compositions* for specific vehicles [21, 22, 75, 76, 77]. Only a proposal by Moti Markovitz and Avishai Wool from the Tel Aviv University attempted a general method for uncovering proprietary CAN payload configurations.

Automated RE of Non-Text Network Protocol Payloads.

As of 2018 the approach proposed by Markovitz and Wool is the only published method found to address the problem of automated RE of non-text based network protocol payloads [8]. Their approach iteratively considers all possible *tokens* t_k for a given set of payloads X_{ID}. Their method first counts the number of unique levels found in each *signal* produced using a particular subset of *bit positions* defined by t_k. The final composition of *signals* $Comp_m(X_{ID})$ is then selected using a greedy approach with the following three phases.

1. **Constant Tokens**: *time series* defined by the set of *bit positions* in t_k with one unique level are considered for inclusion into $Comp_m(X_{ID})$. The *tokens* t_k with a larger bit-width $||t_k||$ are preferred over other 'constant' *tokens* with an intersecting set of *bit positions* $x_j \in t_k$ (satisfying lemma 2). Once all 'constant' tokens have either been added to $Comp_m(X_{ID})$ or removed from the population of candidates generated during the initial brute force *token* generation, all remaining candidate *tokens* with *bit positions* intersecting one or more of the 'constant' *tokens* added to $Comp_m(X_{ID})$ are also removed from the population of candidate *tokens*.

2. **Multi-Value Tokens**: *signals* defined by the set of *bit positions* in t_k with more than one unique level and whose total unique levels are below a heuristic threshold are considered 'Multi-Value' *tokens*. As with 'constant' *tokens*, intersecting *tokens* are added to $Comp_m(X_{ID})$ prioritized by the largest bit width $||t_k||$. All intersecting candidate *tokens* are removed from the population of candidates being considered for inclusion into the *composition* $Comp_m(X_{ID})$.

3. **Signal/Counter Tokens**: All remaining *tokens* are added to $Comp_m(X_{ID})$ using the heuristic $\frac{||unique\ values||^2}{2^{||t_k||}}$. *Tokens* t_k are iteratively added to $Comp_m(X_{ID})$

as 'signal/counter' *tokens* until all *bit positions* $x_i \in X_{ID}$ are included in the final composition $Comp_m(X_{ID})$ (satisfying lemma 2).

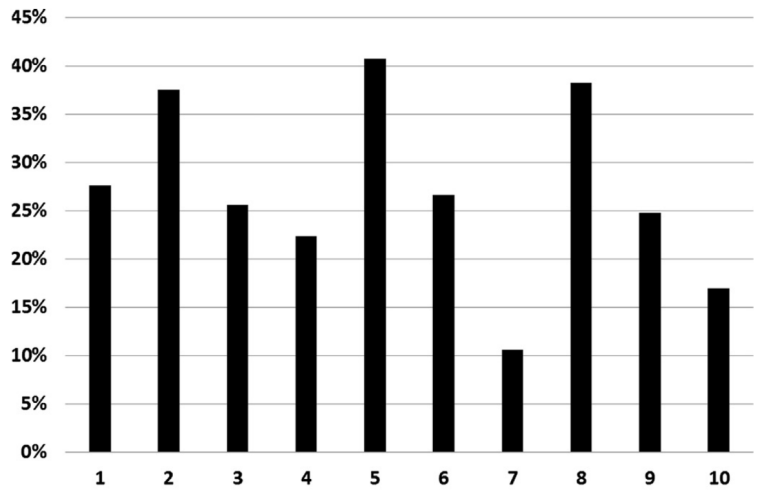


Figure 10. 'Field Classification Accuracy' Validation Results of the CAN Payload Tokenization Method Proposed by Markovitz and Wool [8]

Markovitz and Wool report that this greedy approach had mixed success using a simulated CAN bus containing 10 different arbitration IDs. Fig. 10 show the summary results of the *tokenization* accuracy for 10 simulated arbitration IDs. The horizontal axis is an index of the ten simulated arbitration IDs. The y-axis is a compound metric for tokenization accuracy which includes two factors. First, the accuracy of the algorithm's inferred *composition* compared to the true *composition* forms a score from 0 to 63. Second, each *bit position* $x_j \in X_{ID}$ is scored based on whether it is correctly labeled using one of the three types described earlier. Unfortunately, the exact formula for computing these two factors and combining them is not clearly defined. The method for converting the combined score into the percentage values shown in Fig. 10 is also unclear. Despite not having a precise way for interpreting their results, Markovitz and Wool explicitly note that their payload *tokenization* method performed inconsistently and inaccurately.

The subset selection technique presented in Section 5.2 uses the Shannon Diversity

Index which is notionally similar to the heuristics used by Markovitz and Wool [8, 48]. However, the statistic is used once *tokenization* is already complete. Thus, the pipeline shown in Fig. 11 linking Chapters IV and V is assumed to be the first proposal for robust automated RE of payloads for a non-text network protocol.

Unsupervised and Semi-Supervised Reverse Engineering.

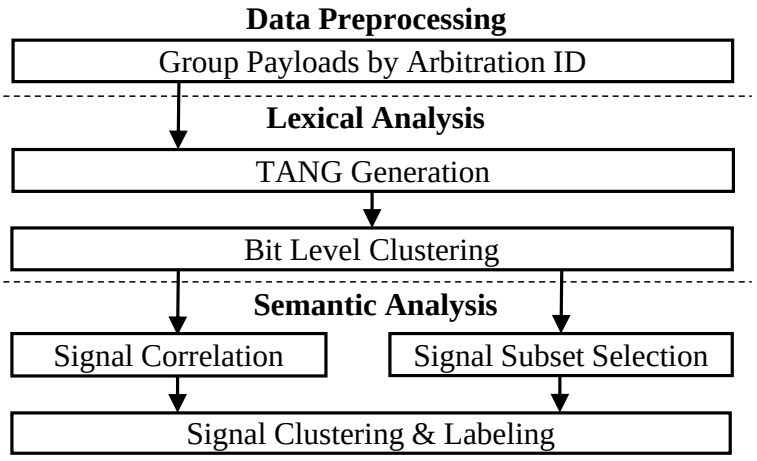


Figure 11. Unsupervised CAN Payload Reverse Engineering Pipeline

Chapters IV and V describe the unsupervised RE pipeline shown in Fig. 11. This proposed pipeline is loosely based on the computer program compiler phases presented in the book *The Theory of Parsing, Translation, and Compiling* by Alfred V. Aho and Jeffrey D. Ullman [29]. The concepts of *lexical analysis*, seperating a piece of data into its logically unique *tokens*, and *semantic analysis*, applying meaning to each *token* produced during *lexical analysis*, are the main ideas taken from that work.

The pipeline also incorporates the concept of *label propagation* from the semi-supervised RE pipeline proposed by Glennan et al. [9]. *Label propagation* refers to the idea of using unsupervised clustering with partially labeled data to produce fully labeled data prior to training a supervised classifier. Examples of supervised algorithms that require labeled training data are k-Nearest Neighbors (k-NN), Support

Vector Machines (SVM), Quadratic Discriminant Analysis (QDA), and Neural Nets [78, 79]. The intent of *label propagation* is to cluster data and copy labels within each cluster to the unlabeled subset in the same cluster. The semi-supervised approach by Glennan et al. achieved *label propagation* using 17 statistical features of *tokens* and the k-Means clustering algorithm [9]. These statistical features are listed in Table 5. Unfortunately, none of the features used by Glennan et al. are viable within the context of constant payload sizes and a non-text multi-cast protocol such as CAN.

Table 5. Feature Set Used by Glennan, Lackie, and Erfrani in Their Semi-Supervised Reverse Engineering Pipeline [9]

Feature Category	Description	# of Features
Bytes (Forwards)	Minimum, maximum, and standard deviation of packets	3
Bytes (Backwards)	Mean, maximum, and standard deviation of packets	3
Inter Packet Time (Forwards)	Minimum, mean, maximum, and standard deviation of inter packet time in the forward direction	4
Inter Packet Time (Backwards)	Mean, maximum, and standard deviation of inter packet time in the reverse direction	3
Duration	Duration of the flow	1
Flag	Whether there was a PSH flag in the forward direction	1
Headers	Total size of the headers in each direction	2

The *label propagation* proposal in Glennan et al. updates unlabeled *tokens* according to the most common labels already attached to *tokens* in the cluster. *Tokens* in clusters with few or no labeled members are assigned the 'unknown' label. This produced a completely labeled data set. The pipeline used by Glennan et al. to achieve *label propagation* is presented in Fig. 12. This semi-supervised pipeline was published

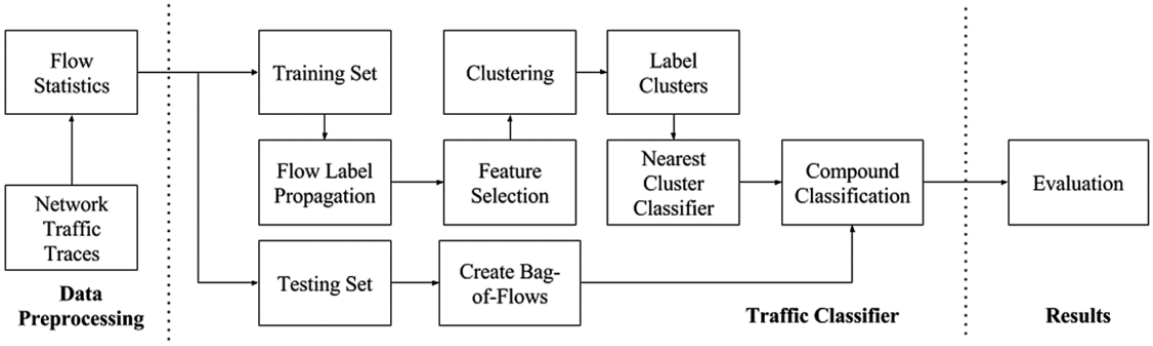


Figure 12. A semi-supervised classifier framework presented by Glennan et al. which is an extension of prior work by Zhang et al. [9, 10]

in 2016 and represents the latest iteration of *label propagation* proposed Erman et al. in 2006 and subsequently extended by Wang et al. in 2011 and Zhang et al. in 2012 [10, 19, 80].

The substantial differences between domains mentioned in this section and Section 2.1 resulted in equally substantial differences between the specific *clustering* implementations. If future research uncovers robust features for measuring *signal* similarity beyond pairwise statistics such as the Pearson's Correlation Coefficient (PCC) or generates a sufficient quantity of labeled data for training a classifier, it may become viable to use the k-means clustering algorithm and a classifier to extend the unsupervised pipeline shown in Fig. 11 to more closely resemble the semi-supervised pipeline shown in Fig. 12.

In the context of this dissertation, the goal of clustering during *semantic analysis* is to apply placeholder labels to approximately continuous numerical *signals* present in a set of CAN payloads X_{ID}. An IDS or extensions to the pipeline shown in Fig. 11 can use truth data such as SAE J1979 diagnostic information, manual classification, or a trained classifier to replace these placeholders with something more semantically meaningful such as "vehicle speed" [20]. Techniques to correctly *tokenize* other types of *signals* is reserved for future work. However, the subset selection and *clustering* techniques presented in Chapter V provide methods for identifying *tokens* that should be re-evaluated. Thus, any future work can extend the pipeline shown in Fig. 11 with another *lexical analysis* phase designed specifically for *tokens* classified as non-continuous data.

Appendix A provides additional discussion about recent automated network traffic RE research. That discussion is ultimately tangential to the work presented in this dissertation but may be useful information for related research.

2.6 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) is an unsupervised clustering "approach which does not require that...a particular choice of K [clusters be selected]. Hierarchical clustering has an added advantage over K-mean clustering [and other alternatives] in that it results in an attractive tree-based representation of the observations, called a *dendrogram*. [...] *Bottom-up* or *agglomerative* clustering...refers to the fact that a dendrogram (generally depicted as an upside-down tree...) is built starting from the leaves and combining clusters up to the trunk" [11].

AHC is particularly well suited for clustering correlated *signals* extracted from a vehicle CAN bus network for the following reasons:

1. There is no requirement to know the number of semantically distinct groups of *signals* within the network *a priori*

2. It is possible to apply the algorithm using only a single pairwise distance measurement such as Pearson's Correlation Coefficient (PCC)

3. Dendrograms provide an intuitive method for understanding how different maximum cluster distance threshold values affect the clustering results

AHC may be most easily introduced by first explaining how to interpret a dendrogram. Fig. 13 from "An Introduction to Statistical Learning" by James et al. provides a small set of 9 observations arrayed in 2-dimensional space according to features X_1 and X_2 and a dendrogram produced using ACH. Looking at the dendrogram on the left, it is possible to determine that that "observations 5 and 7 are quite similar to each other, as are observations 1 and 6. However, observation 9 is no more similar to observation 2 than it is to observations 8, 5, and 7, even though observation 2, 8, 5, and 7 all fuse with observation 9 at the same height, approximately 1.8" [11]. The scatter plot of the same observations on the right hand side of Fig. 13 reinforces these

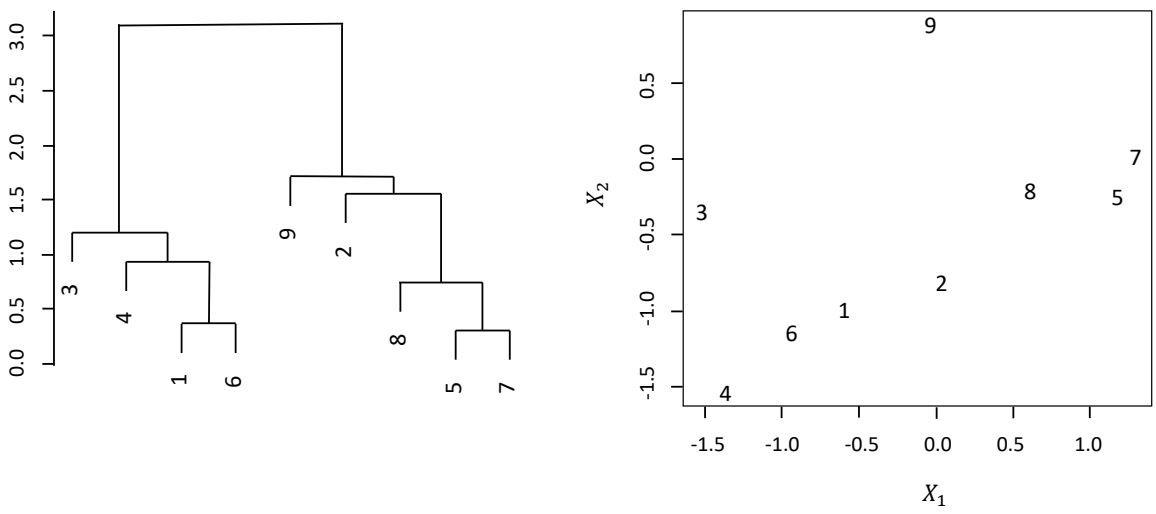


Figure 13. Example Data Set and Dendrogram Produced Using Agglomerative Hierarchical Clustering [11]

style of conclusions made using the dendrogram by observing the similar *euclidean distance* (straight line distance) between observation 9 and observations 2, 8, 5, and 7. Thus, when interpreting dendrograms it is important to understand that the distance between two observations or branches is defined exclusively by the position on the y-axis where they are first merged; horizontal proximity between observations and branches is somewhat arbitrary and **does not** indicate information about similarity.

Fig. 14 is an example of a dendrogram produced from one of the vehicles studied for this dissertation. Each leaf at the bottom of the tree represents a single *signal* extracted using the unsupervised reverse engineering (RE) pipeline introduced in Section 2.5. Clusters are produced by selecting a maximum distance threshold for branches of the dendrogram. Such a threshold may be visualized as a horizontal cut across the tree. In this case, a maximum distance of threshold of 0.2 is reflected by a dashed grey line[5]. Any branches and their leaves below that threshold form clusters. The different red, teal, purple, and green branches reflect 5 different clusters

[5]The threshold value of 0.2 was arbitrarily selected based on subjective expert opinion. It is assumed that some expert insight about the specific OEM or model being analyzed will be needed when selecting this maximum distance threshold.

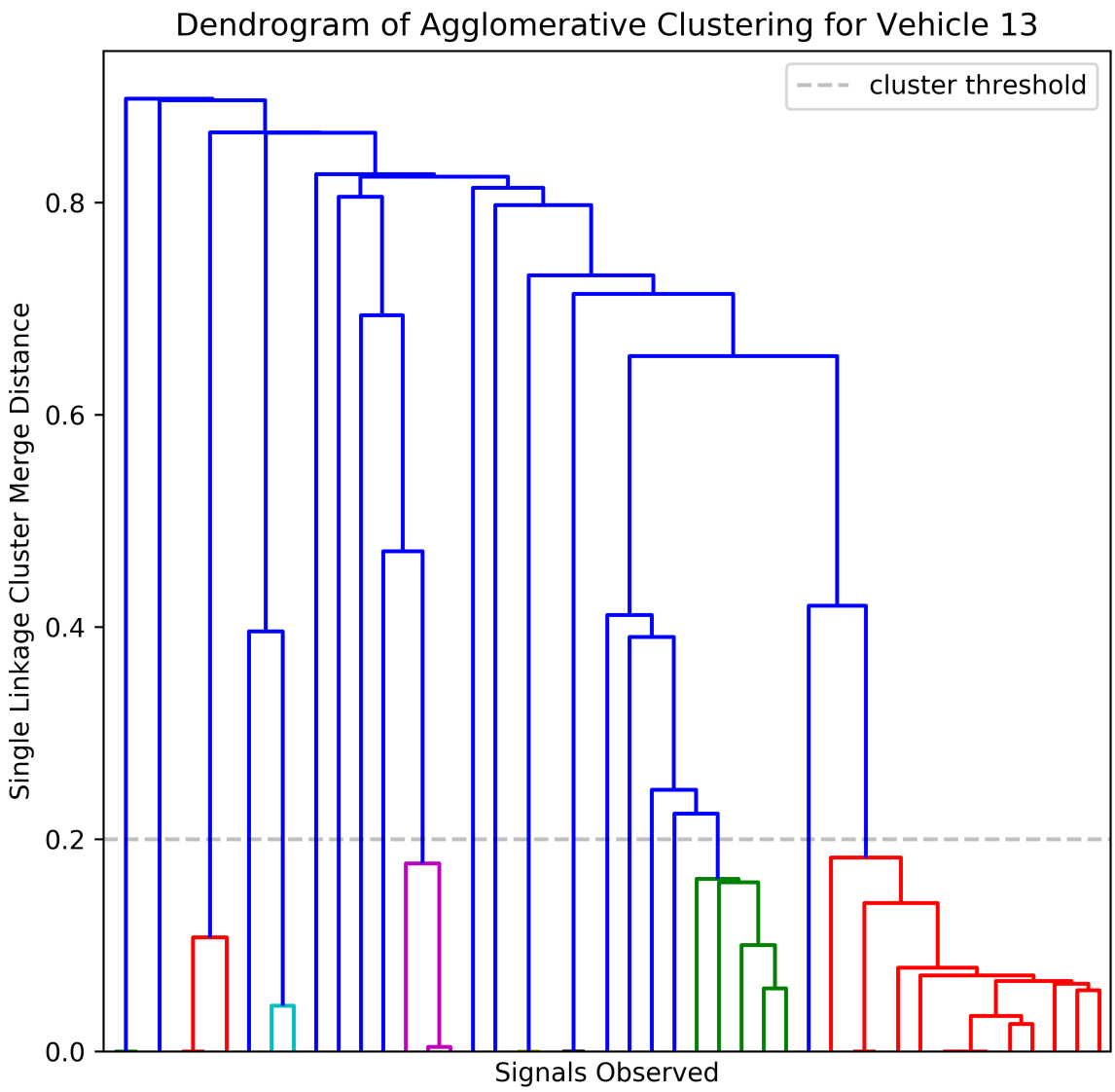


Figure 14. Example Dendrogram of Agglomerative Hierarchical Clustering for a Passenger Vehicle

of *signals* produced using AHC and this particular threshold value. All remaining *signals* marked by blue are left as individual observations.

The actual AHC algorithm is implemented by iteratively considering pairs of observations or clusters. "Starting at the bottom of the dendrogram, each of the n observations is treated as its own cluster. The two clusters that are most similar to each other are then *fused* so that there are now are $n-1$ clusters. Next the two clusters that are most similar to each other are fused again, so that there are now $n-2$ clusters. The algorithm proceeds in this fashion until all of the observations belong to one single cluster, and the dendrogram is complete" [11].

To perform the iterative pairwise similarity measurement, two pieces of information are needed: a *dissimilarity* measurement between each pair of observations and a strategy for measuring the dissimilarity between clusters. The *dissimilarity* metric used to produce Fig. 14 is introduced in Chapter V. The strategy for measuring dissimilarity between clusters is referred to as *linkage*. Table 6 provides a summary of some *linkage* strategies often used with AHC.

2.7 Empirical Data Modeling (EDM)

The field of Empirical Data Modeling (EDM) "is based on the mathematical theory of reconstructing system attractors from time series data" introduced by Floris Takens in a 1981 paper [12, 81]. EDM is focused on modeling nonlinear dynamic systems using observational time series data. The lack of assumptions, focus on observational *time series* data, and largely unsupervised nature make EDM well suited for the automated reverse engineering domain.

Fig. 15 provides a summarized visual explanation for the main ideas involved in Takens' Theorem and EDM. In part (A), the Lorenz attractor is presented as an example of a dynamic system. If observations are made over time using only

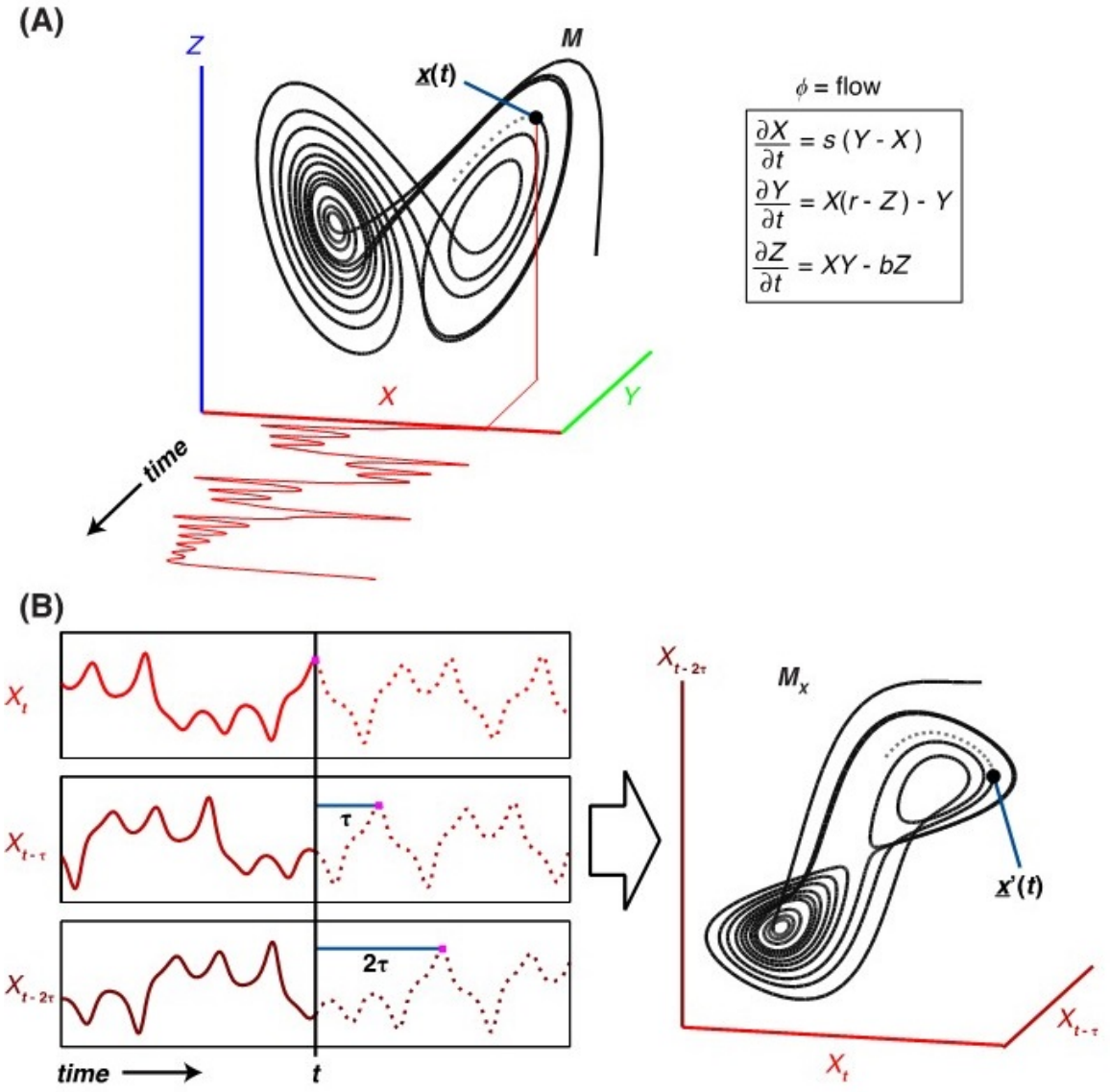
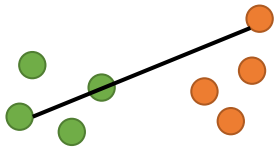
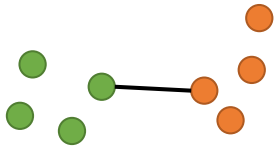
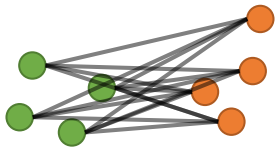
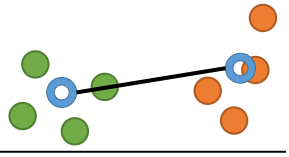


Figure 15. "Empirical dynamic modeling: (A) Example Lorenz system. The attractor manifold M is the set of states that the system progresses through time. Projection of the system state from M to the coordinate axis X generates a time series. (B) Lags of the time series X are used as coordinate axes to construct the shadow manifold M_X which is diffeomorphic (maps 1:1) to the original manifold M. The visual similarity between M_X and M is apparent" [12]

Table 6. A Summary and Visualization of Four Linkage Methods for Agglomerative Hierarchical Clustering

Linkage	Description	Visualization
Complete	Intercluster dissimilarity is the maximum pairwise dissimilarity observed between an observation in cluster A and cluster B.	
Single	Intercluster dissimilarity is the minimum pairwise dissimilarity observed between an observation in cluster A and cluster B.	
Average	Intercluster dissimilarity is the average of all pairwise dissimilarity observed between observations in cluster A and cluster B.	
Centroid	Intercluster dissimilarity is the measured using the centroid for cluster A (the mean of observations) and the centroid for cluster B.	

one dimension of the attractor, then *time series* data is collected. As shown in (B), univariate *time series* data can be converted to a higher dimensional representation by using time lagged versions of itself as additional dimensions. The resulting manifold is referred to as a *shadow manifold* with an example shown in (C) [12]. Takens showed that *shadow manifolds* created in this manner are *diffeomorphic* (map one-to-one) to the original attractor manifold M.

Sugihara et al. demonstrated that the *diffeomorphic* property of *shadow manifolds* can be leveraged to discover whether two *time series* belong to the same dynamic system and are thus causally related [13]. Sugihara et al. showed that the *diffeomorphic* relationship between the original attractor manifold M and each *shadow manifold* means they are also *diffeomorphic* with respect to each other. Thus, if two *shadow manifolds* are shown to be *diffeomorphic* with respect to each other, they may be assumed to belong to the same dynamic system. A technique called Con-

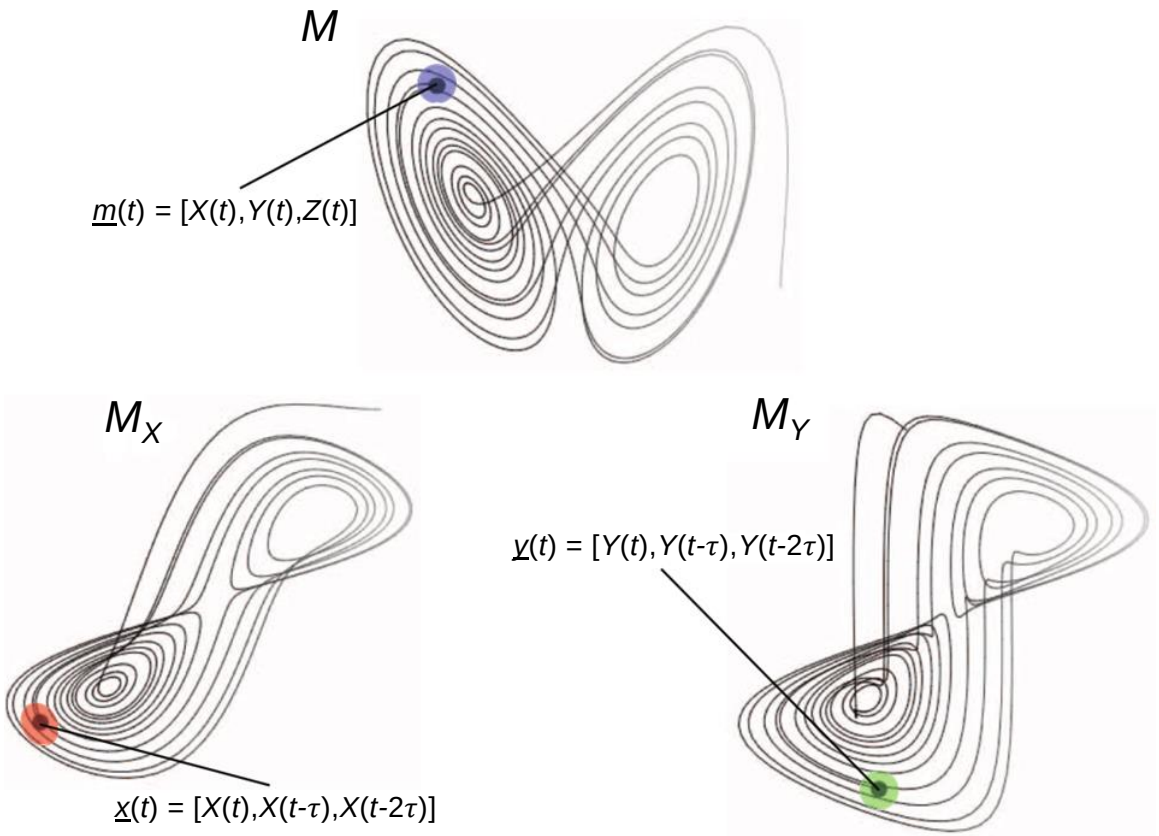


Figure 16. **"Convergent cross mapping (CCM) tests for correspondence between shadow manifolds. This example based on the canonical Lorenz system (a coupled system in X, Y,and Z...) shows the attractor manifold for the original system (M) and two shadow manifolds, M_X and M_Y, constructed using lagged-coordinate embeddings of X and Y, respectively (lag=τ). Because X and Y are dynamically coupled, points that are nearby on M_X (e.g., within the red ellipse) will correspond temporally to points that are nearby on M_Y (e.g., within the green circle). That is, the points inside the red ellipse and green circle will have corresponding time indices (values for t). This enables us to estimate states across manifolds using Y to estimate the state of X and vice versa using nearest neighbors. With longer time series, the shadow manifolds become denser and the neighborhoods (ellipses of nearest neighbors) shrink, allowing more precise cross-map estimates..." [13]**

vergent Cross Mapping (CCM) quantifies the predictive power between two *shadow manifolds* hypothesized to belong to the same dynamic system to infer the presence and magnitude of causality between the *time series* data used to create them [13].

The idea of *cross mapping* is based on testing whether an arbitrary point and its nearest neighbors in one *shadow manifold* can accurately predict a point and its nearest neighbors in another *shadow manifold*. Fig. 16 originally presented by Sugihara et al. provides a visual example of this idea [13]. Sugihara et al. demonstrated that iteratively increasing the *library* (sample) size used to create the respective *shadow manifolds* should result in increased prediction accuracy if they are *diffeomorphic*. They also showed that the increased cross mapping prediction skill with respect to larger *library* sizes should also converge to some maximum limit [13, 82].

In the context of unsupervised mapping of logical relationships in automotive CAN networks, CCM may be very useful for finding important relationships among observed CAN payloads which would otherwise go unnoticed without heuristics or human assistance. The findings in Chapter VI demonstrate that EDM can significantly extend the utility of the unsupervised CAN Payload Reverse Engineering Pipeline introduced in Section 2.5 and discussed in Chapters IV and V. Manifold based predictive modeling also appears to be a viable basis for Intrusion Detection Systems (IDS) for CPS networks. Section 6.5 briefly explores prediction using manifold reconstruction.

The following sequence of EDM techniques is recommended by Ye et al. to best understand and interpret important characteristics of the data:

1. Execute a *train-test* split of each *time series* (*signal*) being studied

2. Perform simplex projection using the *train-test* subsets [83].

    - Determine Embedding Dimensionality E which maximizes forecast skill ρ.

3. Characterize whether the data appears to exhibit deterministic chaos using

simplex projection and E.

4. Characterize nonlinearity of the data using Sequential Locally Weighted Global Linear Maps (S-map) and E [84].

5. Quantify causality with CCM using E and two *signals* to generate *shadow manifolds* to test predictive accuracy [13, 85].

    - Performing CCM using multiple values for the lag parameter τ enables both measurement of the observed strength of a causal relationship and the time delay between cause and effect.

To greatly summarize the technique, simplex projection is the process of iteratively selecting points Y_t in a *shadow manifold* and b other points whose histories over time t are most similar to the currently selected point [17, 83]. A *simplex* is a generalization of a triangle or tetrahedron to an arbitrary number of dimensions [86]. The weighted average of the future values of the b other points is used to make predictions about future values of Y_t. The difference between predictions made in this manner $\hat{Y}_{t+1}$ and the actual future values of Y_t provide a forecast skill ρ. This process can be iteratively repeated for *shadow manifolds* created with different dimensionality to find the embedding dimension E which optimizes ρ.

S-map analysis works in a similar iterative fashion but instead creates linear regression vectors using all neighboring points. The regression vectors are aggregated to approximate an n-dimensional spline. The spline is then compared to the *shadow manifold* attractor to measure ρ [17, 84]. A non-linear tuning parameter θ is used to adjust weighting of neighbors with respect to their distance to the current focal point X_t when generating the regression estimates. If ρ is maximized when $\theta = 0$, then the time series may be assumed to belong to a simple linear system instead of a dynamic system.

Whitey's Theorem asserts that the embedding dimension E "will always lie between the number of variables in the system n and $2n + 1$" [17, 87]. Thus, simplex projection provides insight into the true dimensionality of the dynamic system responsible for generating observation data without requiring complete understanding of the system itself. In the case of vehicle network analysis, understanding the range of dimensionality for a vehicle network may be useful in conjunction with the available *shadow manifolds* for creating high quality models of the vehicular network. Such models could be used to implement effective Intrusion Detection Systems (IDS) for the CAN network. In the context of this paper, knowing E is necessary for accurate application of CCM to detect causality between *signals*. Likewise, the results of S-map analysis indicate whether *signals* relationships common to all vehicles, such as vehicle speed and brake pressure, may be expected to produce linear manifolds and thus be appropriate candidates for analysis using computationally simpler methods such as Granger Causality or auto-regressive linear models [17, 84, 88, 89].

2.8 Verification and Validation

"It is imperative to develop analytical techniques capable of assessing all forms of data, including functional data, to build credibility and confidence" in models and techniques used to solve research questions [90]. Verification & Validation (V&V) is the field of study dedicated to the development and use of such analytical assessment techniques to ensure models and techniques "within [their] domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" or technique [91].

"Verification determines if the model [or technique] is built correctly and validation evaluates whether the correct model [or technique] is being used" [90]. A summary taxonomy of V&V techniques proposed by Dr. Osman Balci from Virginia

Figure 17. A Taxonomy of Verification and Validation Techniques Proposed by Osman Balci [14]

Tech is presented in Fig. 17. This section's discussion and the remainder of this research focuses exclusively on validation techniques. While verification of the greedy techniques proposed in Sections 4.3 and 5.3 may be necessary, their monolithic and relatively simple implementation make it unlikely that major *implementation* errors in the program code occurred. The remainder of the implementations discussed in Chapter III are open-source libraries receiving active open source development. Verifying whether these open source libraries are correct is beyond the scope of this dissertation.

Several validation techniques listed in Fig. 17 do not explicitly require access to truth data. Table 7 presents this subset of validation techniques along with their descriptions proposed by Dr. Andrew Atkinson from the Air Force Institute of Technology [90].

Some combination of comparison testing and sensitivity analysis are used to validate the various data collection and analysis techniques presented in this research. This combination of techniques is well suited for validation in the scope of this dissertation for several reasons. First, comparison testing enables the consistency of input and output be quantified without any truth data. In the context of CAN payload *tokenization*, comparison testing takes the form of quantifying the similarity of a particular arbitration ID's payload *tokenization* $Comp_m(X_{ID})$ between independent driving samples from the same vehicle. By combining comparison testing with sensitivity analysis, it is possible to test assumptions or techniques without truth data.

Varying the driver, route, and duration should not affect the payload *tokenization* if the lemmas presented in Section 2.4 hold true. If the *tokenization* output of the *lexical analysis* techniques proposed in Chapter IV is highly consistent across sets of similar and dissimilar driving samples, this serves as validation that the lemmas

Table 7. A selection of validation techniques which do not require truth data

Validation Technique	Description
	Informal Technqiues
Face Validation	Face Validation is a process in which SMEs review the model and judge whether the behavior and output is reasonable.
	Static Techniques
Cause-Effect Graphing	Cause-Effect Graphing identifies what actions or settings within the model causes which effects in order to assess model correctness.
	Dynamic Techniques
Beta Testing	Beta Testing is the operational testing of the beta version, or the second, revised version of the model. It is usually conducted using realistic conditions.
Comparison Testing	Comparison Testing is used when there are multiple versions of a simulation model that are meant to represent the same system. These versions are executed and then compared against each other.
Functional (Black-Box) Testing	Functional Testing inspects the accuracy of the transformation between the input and output of the model.
Predictive Validation	Predictive Validation uses input and output data from the real system that is being modeled. This input data is used with the simulation model to generate output which is then compared to the system output.
Sensitivity Analysis	Sensitivity Analysis involves changing the input parameters to the model and reviewing how that affects the model output; the nature in which the output changes in relation to the input should mirror that of the system.
	Formal Techniques
Induction, Inference, Logical Deduction	Induction, Inference, and Logical Deduction refer to the act of justifying a conclusion based on some set of given premises. The argument should follow established rules of inference to reach the conclusion.
Lambda-Calculus	Lambda-Calculus transforms the model into formal expressions so that mathematical proof of correctness techniques may be applied.

proposed in Section 2.4 are reasonable and the techniques are consistent. This style of informal inductive reasoning using comparison testing and sensitivity analysis are applied throughout Chapters IV, V, and VI in order to answer the research questions posed in Section 1.2 and validate the lemmas proposed in Section 2.4.

The remainder of this section focuses on enumerating various inputs and outputs to vehicular CAN bus networks and the unsupervised reverse engineering pipeline introduced in Section 2.5 and discussed in Chapters IV and V. Features, metrics or techniques for performing comparison testing of each input and output are introduced.

Brief descriptions of those metrics and techniques are provided; however, extensive discussion about their use in prior work is beyond the intent and scope of this section. Table 8 provides a summary of the features, metrics, and techniques discussed.

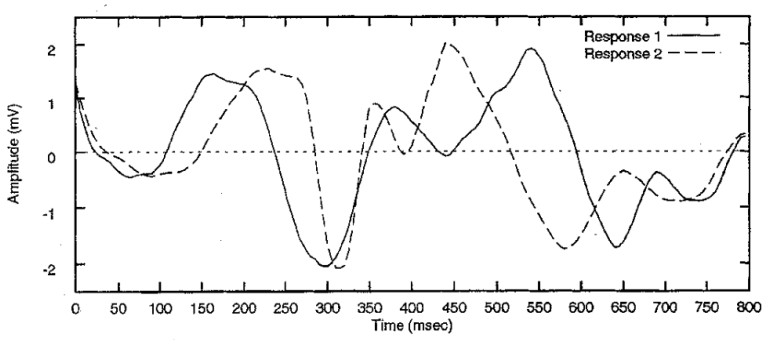
Table 8. Input and Output Features and Metrics Used to Perform Validation

System\Technique	Input Features	Input Metrics	Output Features	Output Metrics
Vehicle Network	Vehicle Speed	Dynamic Time Warping	Set of Arb IDs Set of Payload Matrices	Jaccard Index
Lexical Analysis	*Output from vehicle network*		Set of Payload Compositions Set of Time Series	Alignment Score
Semantic Analysis (Correlation)	Time Series	Jaccard Index	Set of Labeled Clusters	Jaccard Index J1979 Correlation
Semantic Analysis (Causation)	Time Series	N/A	Embedding Dimension Non-Linear Tuning Parameter Manifold Reconstruction(s)	Simplex Forecast Skill S-Map Forecast Skill CCM Forecast Skill Multi-View Forecast Skill

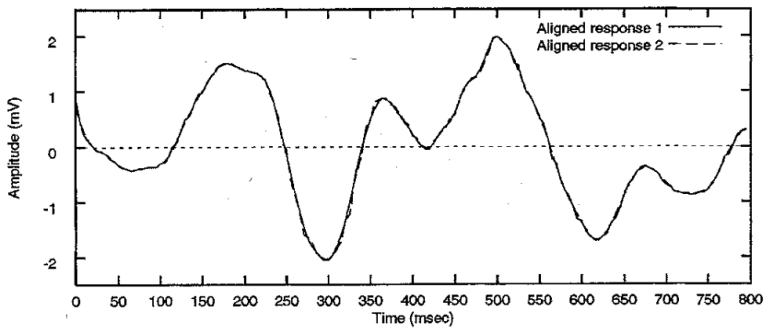
Vehicle Network Input.

Vehicle speed is used for quantifying similarity of input to a vehicle network for four reasons. First, it is agnostic to the mechanical and physical characteristics of individual vehicles. Motorcycles, hybrid-vehicles, fully electric vehicles, etc. all monitor current speed using equivalent units of measurement (distance over time). Second, the findings presented in Section 6.4 support the intuitive assumption that vehicle speed effectively includes information about many other features of the vehicle such as driver input and power-train state. Third, vehicle speed is one of the few J1979 diagnostics which is mandated to be universally supported by all passenger vehicles [20]. This means truth data is available during validation. Fourth, we subjectively believe that focusing on a single feature simplifies the discussion and complexity of comparing input to a vehicle network without significantly changing inferences that may have resulted from using multi-variate alternatives.

Quantifying the similarity between the speed *signals* is possible using Dynamic



Experiment 1: Responses prior to alignment.



Experiment 1: Responses in Fig. 1 after nonlinear alignment.

Figure 18. Example of Non-Linear Signal Alignment Using Dynamic Time Warping [15]

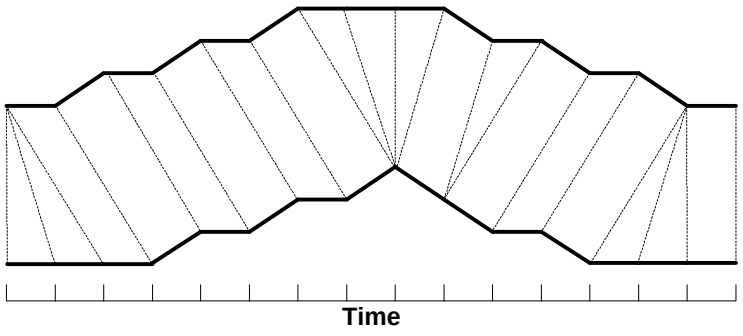


Figure 19. Example Warping Between Two Signals [16]

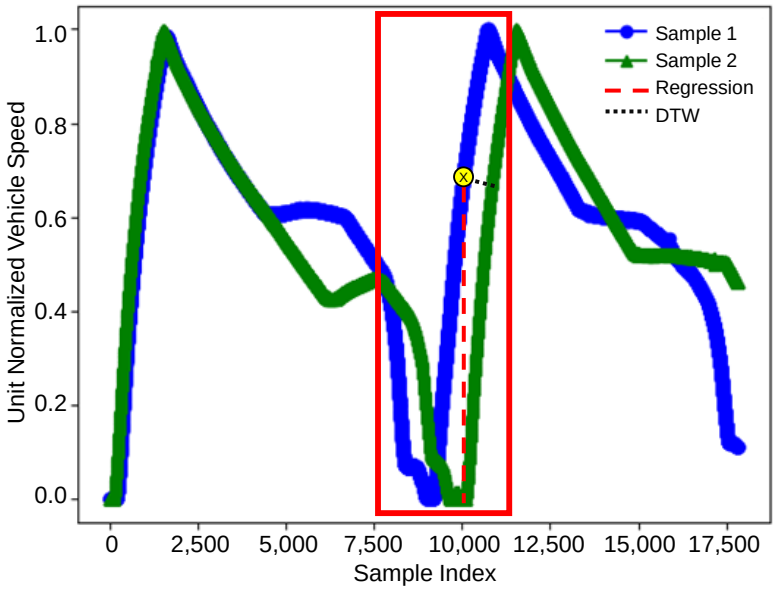


Figure 20. Example of Euclidean Distance Measurement Between Time Series Using Linear Regression and Dynamic Time Warping

Time Warping (DTW). DTW "allows a non-linear mapping of one *signal* to another by minimizing the distance between the two" [15, 92, 93]. Fig. 18 and 19 provide visualizations of this process. Fig. 20 provides an example comparing how distances between two *signals* are measured using linear regression and DTW.

Fig. 20 represents the vehicle speed *signals* from two controlled driving samples collected from the same vehicle. The x-axis is the observed payload index but may be thought of as time. The y-axis represents the vehicle speed over time with higher values corresponding to higher speed. Consider the yellow point on the blue speed *signal* as an example. The vertical dashed red line represents the distance to the green *signal* using standard linear regression. The dashed black line represents the distance measurement to the nearest point to the green *signal* that occurs with DTW. In this specific example, the driver came to a complete stop approximately half way through the driving scenario. However, the moment this full stop occurred is slightly offset by approximately 10 seconds (time is not explicitly shown in the plot). This caused the following hard acceleration and deceleration events immediately following the

stop to also be offset. This example highlights how DTW produces a total distance measurement that is more resilient to this kind of offset in frequency or amplitude that is likely to occur when collecting driving samples from passenger vehicles.

DTW is an $O(n^2)$ time complexity algorithm which makes it difficult to use for larger data sets. Its widespread use in the "bioinformatics, medicine, engineering, and entertainment" domains has led to several versions of DTW exist which improve time complexity to $O(n)$ in some cases [93]. The FastDTW algorithm was chosen for use in this dissertation due to its potential for $O(n)$ runtime and memory complexity and implementation as a Python package [16, 94].

Vehicle CAN Bus Network Output.

The output of a vehicle's CAN bus network may be expressed as a set of matrices X_{ID} which represent chronologically ordered payloads aggregated by arbitration ID. The Jaccard similarity coefficient (Jaccard Index) and conditional logic may be used to quickly quantify the similarity between the set of arbitration IDs observed in different sample of network traffic. This metric has been used as a similarity metric in at least three other automated network protocol reverse engineering proposals [95, 96, 97]. Equation 1 defines the Jaccard Index which is the magnitude of the intersection of two sets divided by the magnitude of their union. If the two sets are identical, then the Jaccard Index is 1. If they are disjoint, the Jaccard Index is 0.

$$J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{1}$$

The Jaccard Index may be used to efficiently calculate the similarity between the set of arbitration IDs observed in two samples of CAN bus traffic. Additional domain specific conditional logic may be used to further refine this similarity measurement. For example, lemma one from Section 2.4 requires that two arbitration IDs are the

same only if their respective matrix of payloads X_{ID} use the same payload bit width n. Statistics about their transmission frequency could also be used in a similar manner. The goal of adding conditional logic is to ensure the lemmas proposed in Section 2.4 are maintained. For the purposes of this research, two arbitration IDs are considered the same if their respective matrix of payloads X_{ID} have the same bit width n. This requirement ensures that any similarity measurements for the IDs output in different samples serve as tests for validating lemma one from Section 2.4: vehicular CAN frames use a consistent payload bit width for each arbitration ID. Specifically, lemma one is considered to be a valid assumption if the Jaccard Index is very large[6] when comparing the sets of observed IDs in several different samples.

CAN Payload Lexical Analysis.

The set of arbitration IDs and their matrix of chronologically ordered payloads X_{ID} serve as the input to the unsupervised *lexical analysis* phase of the automated reverse engineering pipeline shown in Fig. 11 and discussed in Chapter IV. The output of the *lexical analysis* process is a set of payload *token compositions* $Comp_m(X_{ID})$ and the set of time series *signals* extracted using those *compositions*. Comparison testing of the *tokenization* results is possible using the Jaccard Index or the novel *alignment score* metric.

Fig. 21 provides an example comparing two *tokenization* results for a 16-bit pay-

[6]For this context, a *very large* Jaccard Index is assumed to be approximately 0.9 or higher.

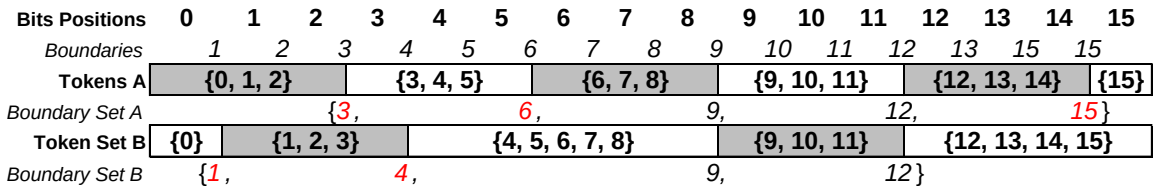


Figure 21. Example of Two Payload Tokenization Compositions for a 16-bit Payload

load. The sets of *token* boundaries are listed as boundary set A and boundary set B. An example calculation of the Jaccard Index using the *tokenization* boundary sets A and B shown in Fig. 21 is demonstrated in Equation 2.

$$J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|} = \frac{|\{9,12\}|}{|\{1,3,4,6,9,12,15\}|} = \frac{2}{7} \approxeq 0.29 \tag{2}$$

We propose that the Jaccard Index masks the fact that these *tokenization* results agree for 10 out of the 15 *bit position* boundary decisions in this example. This discrepancy makes the Jaccard Index less desirable than the novel *alignment score* proposed in Equation 3. The intuition behind the *alignment score* is to evaluate the number of dissimilar boundary decisions with respect to the total number of possible boundary decisions $n-1$; n is the payload bit width. Using the total boundary decisions as a scaling factor allows the metric to be more representative of the true similarity between *tokenization* results. This scaling also allows the metric to be directly compared despite dissimilar payload bit widths.

$$S(A,B) = 1 - \frac{|A \ominus B|}{n-1} \equiv 1 - \frac{|A \bigcup B - A \bigcap B|}{n-1} \tag{3}$$

In Equation 3, the $\ominus$ operator represents the *symmetric difference* of sets A and B which is their intersection $A \bigcup B$ minus their union $A \bigcap B$ [98]. The *symmetric difference* of boundary sets A and B is the set of dissimilar boundary decisions. Equation 4 demonstrates a calculation of the *alignment score* for the example shown in Fig. 21. The range of the *alignment score* is the same as the Jaccard Index with 1 corresponding to identical *tokenization* output. An *alignment score* of 0 results from a pair of *tokenization* results which do not share any matching bit boundary decisions.

$$
\begin{aligned}
S(A,B) &= 1 - \frac{|\{1,3,4,6,9,12,15\} - \{9,12\}|}{16-1} \\
&= 1 - \frac{|\{1,3,4,6,15\}|}{15} \\
&= 1 - \frac{5}{15} = \frac{2}{3} \approxeq 0.67
\end{aligned} \tag{4}
$$

In the case of one *tokenization* result representing truth data, as is the case with the simulated payloads used by Markovitz and Wool and the simulations discussed in Section 3.3, this *alignment score* metric serves as a measure of accuracy for the experimental *tokenization* result [8]. In the more likely scenario where neither *tokenization* result is known to be correct, as is the case with comparison testing using the same vehicle in different driving conditions, this metric serves as a measure of consistency.

Time Series Semantic Analysis: Unsupervised Labeling.

The unsupervised *semantic analysis* phase of the unsupervised reverse engineering pipeline introduced in Section 2.5 and detailed in Chapter V uses the *signals* produced by the *lexical analysis* phase described in Chapter IV. While the *alignment score* metric quantifies how similar payload *compositions* are between driving samples, adding or removing a single *bit position* x_j comprising a particular *signal* may drastically alter its numerical interpretation. Calculating the Jaccard Index of the set of *tokens* t_k output from *lexical analysis* provides a quantitative baseline for how similar the input to *semantic analysis* is between the two samples. In this context, the intersection is the subset of *tokens* t_k which are *exactly* the same set of *bit position* x_j extracted from a particular arbitration ID's set of payloads X_{ID}. All other *tokens* t_k are therefore members of the *symmetric difference* of the two sets of *tokens* output during *lexical analysis* of the respective CAN bus data samples.

The output of the *semantic analysis* discussed in Chapter V is labeled clusters

of *signals*. The Jaccard Index can quantify whether these clusters represent similar groups of *signals*. Calculating the Jaccard Index between each potential pair of clusters can provide insight into intra-sample and inter-sample relationships in a fashion similar to a correlation matrix. For example, a large Jaccard Index between two clusters serves as a measure of the similarity between the exact *tokens* (define by Arb ID, left most bit position, and right most bit position) comprising each cluster. If those two clusters represent the same vehicle in different driving conditions, a large Jaccard Index may be interpreted as validation that the *semantic analysis* method is robust to variations in driving conditions. If the two clusters belong to samples from different vehicles, the Jaccard Index may reveal whether the OEMs manufactured the vehicles with similar sets of Arbitration IDs, payload *compositions*, and relationships to physical processes within the vehicles. As a domain specific validation technique, it is also interesting to perform correlation between J1979 diagnostic data and *signals* which were clustered during *semantic analysis*. Calculating the Jaccard Index of *signals* assigned a particular label using J1979 data compared to *signals* that did not get the same label within individual clusters provides additional insight into the homogeneity of *signal* clusters produced during *semantic analysis*.

Time Series Semantic Analysis: Empirical Data Modeling.

Validation metrics and techniques relevant to the Empirical Data Modeling (EDM) were introduced in Section 2.7. Those concepts are discussed further in Chapter VI.

III. Research Method

3.1 Assumptions and Limitations

There are several overarching assumptions and limitations regarding the unsupervised reverse engineering pipeline introduced in Section 2.5 and discussed in Chapters IV and V. This section presents those general assumptions and limitations. Other assumptions or limitations specific to the topic of a particular chapter appear after each chapter's introduction.

Sample Similarity.

Comparison testing and sensitivity analysis are used for evaluating whether the research questions posed in Section 1.2 are answered by the approaches in Chapters IV, V, and VI. There is no way to completely control or fully predict the output of passenger vehicle CAN bus networks. Factors such as individual ECU hardware clock drift, CAN bus arbitration resolution, minor or major variations in driver input, mechanical component performance fluctuating within operational tolerance, minor or major changes to driving route, duration, and conditions, and data logging errors or omissions all affect what ultimately appears in a particular sample of CAN bus network traffic. The following three categories of sample similarity are assumed to be sufficient for the purposes of performing validation using a combination of comparison testing and sensitivity analysis.

- Samples are considered to be *very similar* when collected using the same vehicle operated by the same driver using a scripted driving scenario on the same length of paved roadway which is free of obstacles. The two vehicle speed *signals* shown in Fig. 20 from Section 2.8 are examples of *time series* from *very similar* samples.

- Samples are considered to be *similar* when collected using the same vehicle operated by the same driver using a scripted driving scenario using approximately the same length of paved roadway with approximately the same level of vehicle traffic along the route.

- Samples are considered to be *dissimilar* when collected using the same vehicle operated without a scripted driving route with major variations in traffic patterns, traffic control devices, or speed limits. An example of a major variation is comparing a sample collected during 'heavy city traffic' to a sample predominantly collected at freeway speeds with few traffic control devices. A subjectively large difference in sample duration is another example of a major variation.

Vehicle Network Output.

The lemmas regarding constant payload bit width and logical *composition* described in Section 2.4 are assumed to hold true regardless of major variations to the sample collection conditions. Thus, any variation in payload *tokenization* accuracy or consistency is assumed to be exclusively caused by shortcomings of the *lexical analysis* strategy.

Limited Empirical Validation: Truth Data and Network Simulation.

Section 2.3 discussed how access to OEM generated DBC files are necessary to correctly interpret the CAN bus network traffic for a particular vehicle. The limited distribution of these files means validation presented in this dissertation and similar work in the automotive network reverse engineering domain relies upon simulations to generate truth data. As of 2018 there is no known published research regarding the probability distribution of a particular payload *composition* being used by a par-

ticular manufacturer, vehicle model, model year, or vehicle type. There is also no published research regarding the distribution of arbitration IDs, payload bit widths, *signal* types, or other features important to producing simulation models that accurately represent real world CAN bus networks.

A 64-bit CAN payload has $2^{64-1} = 9,223,372,036,854,775,808$ possible *compositions*. Markovitz and Wool assumed that each *token* within each of those *compositions* can be one of three *signal* types [8]. Each payload *composition* belongs to one of 2^{11} (standard frame format) or 2^{29} (extended frame format) possible arbitration IDs. Those arbitration IDs in turn belong to some combination of $\frac{n!}{k!(n-k)!}$ arbitration IDs comprising a vehicle's network where n is the maximum IDs allowable by the frame format (2^{11} or 2^{29}) and k is the number of IDs actually used.

The exponentially large space of possible vehicle CAN bus network configurations combined with no existing research models implies that simulations of vehicle network payloads and *signal* behavior are effectively arbitrary. The approaches in this dissertation are ultimately intended to improve the frequency and accuracy of CAN bus network modeling research; however, some kind of simulation strategy is needed to validate the *lexical analysis* approach. Thus, the simulation strategy proposed in Section 3.3 and used for validation in Chapter IV is arbitrary but assumed to be reasonable based on subjective expert opinion. The use of Monte Carlo simulation techniques is intended to partially mitigate the arbitrary nature of the simulations from a validation standpoint. However, the random data generation methods and their parameters are also chosen based on subjective expert opinion. Simulation is not used in Chapters V and VI because J1979 diagnostic information and manual reverse engineering could be used for truth data.

3.2 Data Collection

Data collection for testing the proposals in this dissertation took place in two phases. Both data collection phases used the collection device shown in Fig. 22 connected to the vehicle's On-Board Diagnostics (OBD-II) interface[1]. The first phase included a broad study of 17 vehicles representing 8 different Original Equipment Manufacturers (OEMs), 8 model years (2008-2017 except 2013 and 2016), five vehicle types (two and four door sedans, SUV, truck, wagon), hybrid and conventional power trains, and manual and automatic transmissions. Different drivers operated the vehicles using a *similar* scripted driving route. No J1979 diagnostic data was collected during this data collection phase to ensure no logical modification to the vehicle networks occurred.

[1]Technical details about this data collection device are provided in Section 3.4

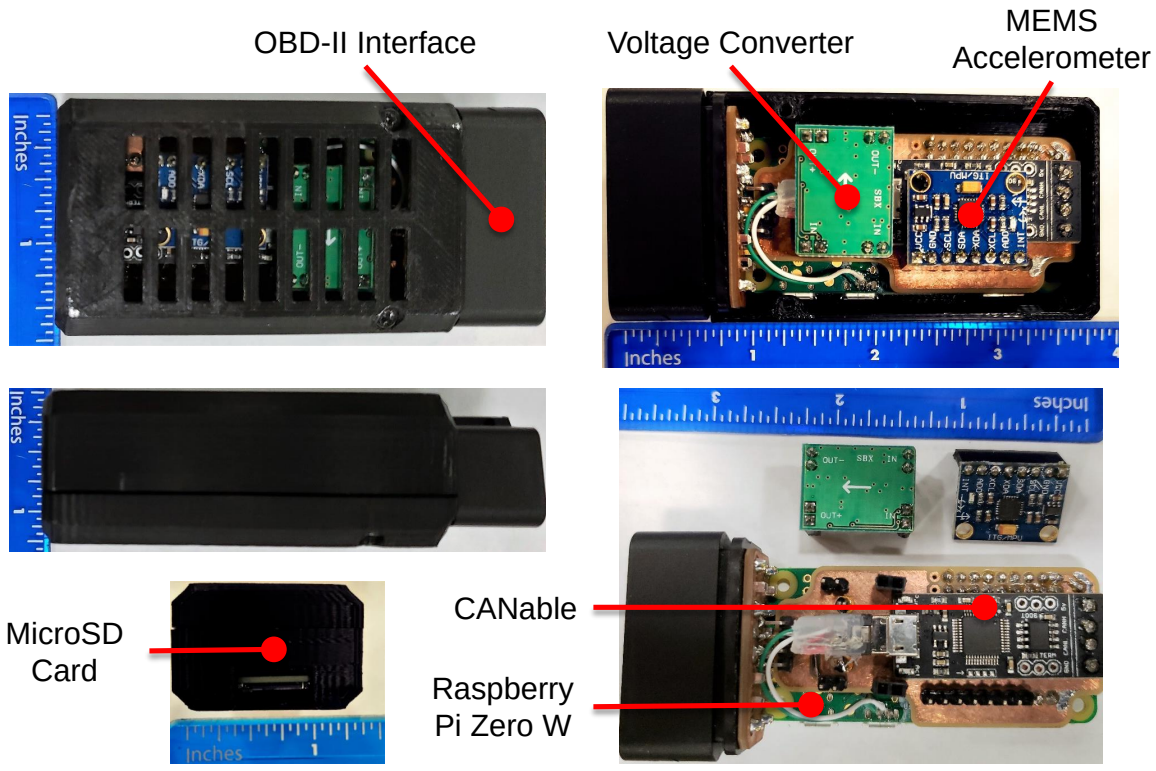


Figure 22. On-Board Diagnostics (OBD-II) Vehicle Network Sniffer

The second data collection phase used only one driver and vehicle. Various combinations of *very similar*, *similar*, and *dissimilar* data samples were collected with the benefit of requesting J1979 diagnostic information during each sample. The *very similar* samples were collected by driving on a 1.3 mile straight closed course with a start and turn around point clearly marked. This controlled driving scenario began with the driver started at a full stop, accelerated to 75 miles per hour (mph), then released all driver input except the steering wheel. After a delay of approximately 30 seconds, the driver began applying modest accelerator and brake pedal input to reach the end of the 1.3 mile straight line closed course. Once near the opposite end, the driver performed a u-turn and came to a complete stop facing the starting position. This sequence of events was repeated until the driver returned the vehicle to the original starting position and orientation. Five samples for this controlled driving scenario were collected in rapid succession. The goal of this collection strategy is to minimize variation of as many environmental, vehicle, and driver related factors as possible. Such factors include road traction, temperature, driver alertness and idiosyncratic behavior, and the vehicle's mechanical performance.

A 'city' driving sample was collected during a four minute drive through light traffic on curved and straight roads that featured various traffic control devices. A second 'city' driving sample consisted of an approximately 30 minute drive through light and heavy traffic and included segments of highway driving. Fig. 23 presents examples of vehicle speed *signals* from each of these three driving scenarios. The x-axis of each figure is time in seconds. The y-axis is the unit scale normalized magnitude of vehicle speed.

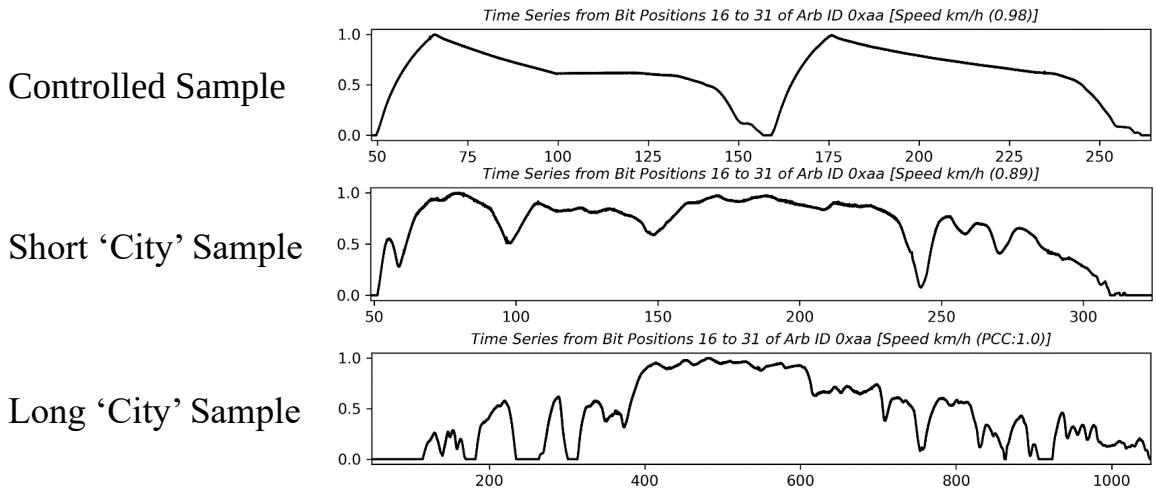


Figure 23. Examples of Vehicle Speed Time Series from Three Dissimilar Driving Scenarios

3.3 Data Simulation

Simulation is used in Chapter IV to augment findings produced using data collected from passenger vehicle CAN networks. Section 3.1 discussed the limitations of simulating CAN networks. To briefly reiterate that discussion, the simulation strategy proposed in this section as well as that in all other academic research pertaining to CAN and similar protocols is effectively arbitrary and based solely on subjective expert opinion.

The *time series* and payload simulations strategies presented in this section represent an attempt to reasonably simulate CAN payloads produced by passenger vehicles. Analysis of whether the proposed simulation strategy is reasonable, how accurately it resembles one or more passenger vehicle CAN networks, and similar research questions are beyond the scope of this research. That said, this research is expected to directly enable future research intended to answer such questions. It is the earnest hope of the author that the techniques presented here helps motivate research in the area of modeling and simulating CAN networks.

Two independent steps form the basis for simulating CAN network data in this proposal. The first step is generating many continuous numerical and non-continuous *time series* using random walks. The second step randomly concatenates those simulated *time series* into a matrix of payloads X_{sim} with an approximate bit width $n \approx 64$. The continuous numerical random walks were generated using a Weiner Process (Brownian Motion) with drift. Equation 3.3 presents the auto-regressive random walk calculation method. The data used to generate a mean and standard deviation for this equation was a vehicle speed *signal* extracted from a 'city' driving sample collected from a real vehicle. The random error term ε is draws from the standard normal distribution $\varepsilon \sim N(0, 1)$.

$$drift = \overline{X} - \frac{Var\left[X\right]}{2} \tag{5}$$

$$x_t = x_{t-1} \times e^{(drift + \sigma \times \varepsilon)} \tag{6}$$

Non-continuous *time series* are generated by adding or removing an arbitrarily 'large' integer value from the current value at a set interval of time. Values are added until the maximum unsigned integer value for the particular bit width being used is reached. Then values are subtracted until doing so would reach 0. This behavior continues to produce an approximately sinusoidal behavior with limits at 0 and 2^n. This non-continuous simulation was performed for bit widths between 2 and 10. The intent for this simulation method is to generate *time series* which are constant except for arbitrarily large gaps between some samples. Again, these *time series* generation and concatenation methods to produce simulated matrices X_{sim} are based on subjective expert insight that they reasonably approximate data generated

by production passenger vehicles; further research in simulating CAN bus networks is needed but beyond the scope of this research.

Fig. 24 presents an example of 100 continuous and 9 non-continuous *time series* generated using the random walk techniques described above. The Transition Aggregation N-Gram (TANG) for each *time series* were independently calculated and then concatenated until a row vector of at least 64 *bit positions* x_i was created[2]. Three different concatenation strategies were used to generate CAN 'payloads' X_{sim} in this manner. The first strategy is to create payloads using only continuous *time series* data by sampling without replacement the total population of simulated continuous *time series*. Once the last concatenation increased the 'payload' bit width to 64-bits or greater, it was saved and a new 'payload' was started. The second strategy is the same as the first except non-continuous *time series* were randomly sampled with replacement and placed between each continuous *time series*. The third strategy used only non-continuous *time series* to generate 100 simulated 'payloads'. The true *composition* $Comp' X_{sim}$ of each X_{sim} generated using these strategies was recorded for comparison during validation.

3.4 Technical Implementation

The CAN bus data collection device shown in Fig. 22 uses a Debian Linux distribution called 'Raspbian' tailored for the Raspberry Pi Zero W hardware. Raspbian release 2017-03-02 is the specific version of the operating system (OS) used to operate the Raspberry Pi Zero W. This OS is a customized implementation of Debian Linux "Jessie" Version 8 based on Linux kernel version 4.9 and compiled using GCC 4.9 [99]. The data logging and J1979 diagnostic polling were accomplished using a combination of version 20161220 of the SocketCAN (https://github.com/linux-can/can-utils)

[2]TANGs are introduced in Section 4.2

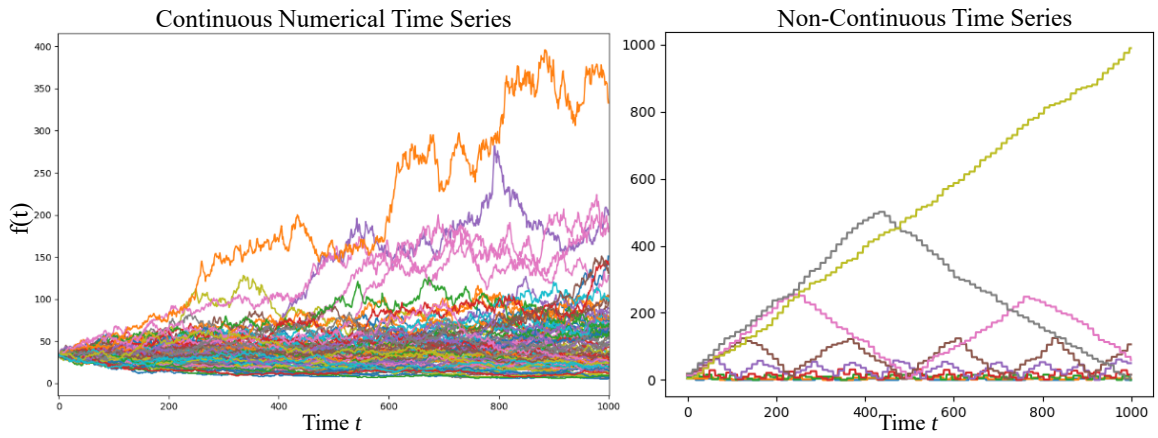


Figure 24. Example Random Walks of 100 Continuous (lefthand) and 10 Non-Continuous (righthand) Time Series

Linux kernel module and a custom activation program written in C++. The custom C++ program made API calls to SocketCAN to request J1979 diagnostic information every 0.25 seconds; however, in practice the vehicle studied using J1979 diagnostics only responded to those requests about once every 10 seconds. Regular CAN traffic were logged to a Comma-Separated Values (CSV) text file using interrupts triggered by SocketCAN and the CAN controller's firmware.

The Raspberri Pi Zero W uses a 1GHz single core CPU and the 'Raspbian' operating system was not designed to provide 'real-time' guarantees to running processes [100]. These limitations make it highly likely that one or more CAN frames transmitted to the OBD-II port by the vehicle's CAN bus were not successfully recorded in the CSV log of observed network traffic. However, an average of 403,221 payloads were observed across the short 'city' driving samples collected from the 17 vehicles studied. This large number of observations is assumed to mitigate any loss of fidelity caused by limitations in the network logging device. Data observable from the OBD-II interface were assumed to be an acceptably representative and faithful representation of data generated by CPS connected to the vehicle's CAN bus.

All techniques related to the unsupervised reverse engineering pipeline introduced

in Section 2.5 and discussed in Chapters IV and V were implemented in Python 3.6. Elements of the Numpy v1.14, Pandas v0.22, and scikit-learn v0.19 packages were used for data structures, interpolation, and normalization [101, 102, 103]. The FastDTW v0.3.2 package was used for its implementation of the FastDTW algorithm [16, 94]. SciPy's matplotlib plotting package was used to produce figures [104].

Chapter VI leverages the rEDM v0.7.1 package developed for R by Ye et al. [89]. It is used to produce all EDM findings discussed in Chapter VI using *signals* generated by the *tokenization* methods described in Chapter IV. The package provides a straightforward set of functions for performing EDM, has two detailed user manuals written be Ye et al. and Chang et al, and its developers include Dr. George Sugihara, the creator of most of the techniques rEDM implements [53, 85].

Data Cleaning.

Data cleaning and pre-processing was routinely performed to implement the techniques presented in this research. This involved interpolating univariate *time series* data to a shared time axis and normalization of magnitudes to unit length magnitude. Interpolation was performed using the `reindex` function from Pandas with the nearest point interpolation strategy. Normalization was achieved using scikit-learn's `minmax_scale` method and its default feature scaling range of 0 to 1. Lastly, arbitration IDs whose set of payloads X_{ID} did not use a constant bit width (Lemma 1 from Section 2.4) or represented a repetition of one value were ignored. Across the 18 passenger vehicle studied, there was an average of less than one arbitration ID which violated lemma 1. It appears that the instances of non-constant payload bit width were primarily caused by data collection errors. In particular, disconnecting the data collection device while it was writing the contents of a payload to the logging file resulted in one incorrectly recorded payload to appear at the end of the log.

IV. Unsupervised Lexical Analysis of CAN Payloads

The *lexical analysis* strategy for the pipeline introduced in Section 2.5 focuses on approximately continuous numerical *signals*. A *signal* was defined in Section 2.4 as a univariate *time series* generated by a cyber-physical system (CPS). Continuous numerical *signals* can and should be *tokenized* before non-continuous *signals* because there is a predictable relationship among the transition frequencies of each *bit position* $x_j \in X_{ID}$ when base-10 (decimal) *signals* are converted to base-2 (binary) representations. This relationship holds regardless of whether two's compliment, one's compliment, signed magnitude, or unsigned encoding is used[1]. The predictable relationship is bit ordering from a *least significant bit* (LSB) to a *most significant bit* (MSB). The LSB represents the 2^0's place while the MSB is the 2^{n-1}'s place where n is the bit width being used.

Empirical analysis of the vehicles discussed in Section 4.4 suggest that CPS sampling analog processes such as velocity or pedal position many times a second produces approximately continuous numerical *signals*. This behavior is also likely to occur with sensors monitoring the joint angles of a robotic arm or the energy output of an x-ray machine. As an example, the vehicle sensors measuring locomotion typically produce *signals* which have small differences between sequential samples. Engine RPM should not change from 1,215, 7,031, back down to 2,580, and then 5,111 within one second. Rather, a predictably smooth change from one value to another occurs between sequential measurements such as 2,000 to 2,031 and then 2,045 RPM. This produces approximately continuous numerical *time series token* t_k with *bit positions* x_j that transition more frequently the closer $x_j \in t_k$ is to the LSB of t_k. The *lexical*

[1]Signed magnitude encoding is the only exception to the predictable relationship among *bit positions*. Signed magnitude encoding is the same as unsigned except the MSB represents the numerical sign. This causes that MSB to transition somewhat independently to the neighboring *bit positions*.

analysis techniques in this chapter are based upon searching for the predictable and approximately monotonic increase of transition frequency of $x_j \in t_k$ from MSB to LSB.

4.1 Assumptions and Limitations

The lemmas discussed is Section 2.4 introduced and motivated the assumptions that each arbitration ID observed on a vehicle CAN bus X_{ID} uses the same *bit width* n and *token composition* $Comp_m(X_{ID})$. This chapter also assumes that the majority of *signals* embedded in a set of vehicular payloads X_{ID} is approximately continuous numerical *signals*. This section also assumes that the majority of sequential samples of a signed numerical *signal* $X_{ID,i}$ do not represent a transition from a positive to a negative value. Such 'crossing zero' incidents are expected to represent a minority of the observed *signal* behavior. *Signals* which do not meet this assumption may cause what is referred to as the 'frequent zero crossing problem.'

Frequent Zero Crossing Problem.

One predictable flaw in the method presented in this chapter is the scenario where there are very frequent 'crossing zero' events of a signed number using one or two's compliment. To explain, imagine a sensor sampling the vehicle's steering wheel angle where the neutral position represents 0. Steering wheel positions counter-clockwise relative to neutral are represented by increasingly negative numbers and positions clockwise from neutral are positive numbers. If the CAN network sample collected while driving exclusively with the steering wheel angle mostly neutral with many small counter-clockwise and clockwise corrections, then the bit positions near the MSB transition nearly as frequently as those near the LSB. This kind of data collection scenario is possible from freeway driving and other places where straight line driving

conditions are common.

The *controlled* driving scenario described in Section 3.1 involved mostly straight line driving conditions where this problem may have occurred. It is not possible to definitively identify, test, or quantify how significant the 'zero crossing problem' was during these samples without a DBC file for the vehicle studied or some other truth data. Thus, detailed analysis of this limitation and research about how to overcome it is beyond the scope of this dissertation.

4.2 Transition Aggregation N-Gram (TANG)

A *bit position* x_j is considered to have transitioned when it flips between 1 and 0 in chronologically-sequenced CAN payloads $X_{ID,i}$ and $X_{ID,i+1}$. Bit level transition analysis can be efficiently calculated by storing observed payloads into an $I \times J$ boolean matrix. I is the number of row vectors with one row per observed CAN message payload $X_{ID,i}$. J is the *bit width* n of the payloads with column vectors representing the relative *bit positions* $x_j \in X_{ID}$. See Table 9 for an example of a 10 x 8 boolean array representing 10 samples of an 8-bit payload.

Table 9. Example Boolean Matrix for an Arbitration ID's Payloads

Observation	Bit Position							
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1

By performing an exclusive or (XOR) of each sequential pair of row vectors $X_{ID,i}$

and $X_{ID,i+1}$ in such a boolean matrix, a *transition matrix* is then created with $I-1$ rows and 1s anywhere a bit transition occurred. Table 10 is the *transition matrix* produced from Table 9. In this example, the 0^{th} row vector $X_{ID,0}$ is XORed with the 1^{st} row vector $X_{ID,1}$.

$$\begin{array}{cccccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \oplus & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$X_{ID,1}$ is XORed with $X_{ID,2}$ and so on for all sequential row vectors in the boolean matrix. Summing the 1s in each column vector (*bit position* x_j) of the *transition matrix* produces a $1 \times J$ row vector. For the remainder of this research, this row vector is referred to as a Transition Aggregation N-Gram (TANG).

$$\begin{array}{cccccccccc} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \oplus & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Table 10. Example Transition Matrix and Transition Aggregation

XOR Result	**Bit Position**							
	$\mathbf{2^7}$	$\mathbf{2^6}$	$\mathbf{2^5}$	$\mathbf{2^4}$	$\mathbf{2^3}$	$\mathbf{2^2}$	$\mathbf{2^1}$	$\mathbf{2^0}$
Observation $0 \oplus 1$	0	0	0	0	0	0	0	1
Observation $1 \oplus 2$	0	0	0	0	0	0	1	1
Observation $2 \oplus 3$	0	0	0	0	0	0	0	1
Observation $3 \oplus 4$	0	0	0	0	0	1	1	1
Observation $4 \oplus 5$	0	0	0	0	0	0	0	1
Observation $5 \oplus 6$	0	0	0	0	0	0	1	1
Observation $6 \oplus 7$	0	0	0	0	0	0	0	1
Observation $7 \oplus 8$	0	0	0	0	1	1	1	1
Observation $8 \oplus 9$	0	0	0	0	0	0	0	1
TANG	**0**	**0**	**0**	**0**	**1**	**2**	**4**	**9**

4.3 Composition Selection Using TANG

The proposed *composition* selection strategy assumes the $x_j \in X_{ID}$ with the largest transition count in a TANG is the LSB of a numerical *signal* $t_k \in X_{ID}$. This LSB should represent a local maximum in the TANG. A contiguous subset of *bit positions* on the left-hand or right-hand side of the local maximum should form a decreasing gradient of transition counts until reaching a local minimum. That local minimum is assumed to be the MSB of the same *token* $t_k \in X_{ID}$ as the LSB. This behavior is demonstrated in the TANG produced from Table 9. *Bit position* 2^0 is the LSB of the *token* representing an unsigned integer sequence counting from 0 in row 0 to 9 in row 9. A decreasing gradient of transition counts extends on the left-hand side of the LSB until reaching a local minimum at the 2^4 *bit position*. That local minimum can optionally be interpreted to extend to the 2^7 *bit position*.

Algorithm 1 presents a greedy search strategy for clustering *bit positions* $x_j \in X_{ID}$ into a composition $Comp_m(X_{ID})$ of *tokens* t_k defining each continuous numerical *signal* concatenated within the set of payloads X_{ID}. Algorithm 1 may be thought of as gradient descent and gradient ascent implemented as a univariate unsupervised clustering method. Gradient descent is a machine learning technique intended to find a local minimum of a function forming a surface or curve; gradient ascent finds a local maximum [105]. The TANG serves as the function in this case with local minima assumed to represent the MSB of each $t_k \in X_{ID}$ and local maxima corresponding to the LSBs. Clustering is performed by grouping $x_j \in X_{ID}$ into disjoint sets referred to as *tokens* t_k. Each t_k represents a contiguous subset of $x_j \in X_{ID}$ which include a local minimum, a set of 0 or more intermediary *bit positions* with a monotonically increasing or decreasing gradient of transitions counts, and a local maximum. Tokens t_k are added to the *composition* $Comp_m(X_{ID})$ until all $x_j \in X_{ID}$ are part of one and only one $t_k \in Comp_m(X_{ID})$ (lemma 2).

Require: A $1 \times n$ **TANG** array. **TANG** index j corresponds to *bit position* $x_j \in X_{ID}$. **TANG** values are the observed transition counts of $x_j \in X_{ID}$.

```
comp: list ← [ ]
left_bit, last_bit: integer ← 0
token_started, endian, padding: boolean ← false
max_inversion: float ← 0.0
for j = 0 to j = n − 1 do
  if TANG[j] = 0 and not padding then
    if token_started then
      comp.append((left_bit, j-1))
      token_started ← false
      left_bit ← j+1
      last_bit ← TANG[j]
    end if
    go to next iteration
  end if
  if token_started then
    if TANG[j] ≥ last_bit and endian then
      {continue this gradient ascent; do nothing for now}
    else if TANG[j] ≤ last_bit and not endian then
      {continue this gradient descent; do nothing for now}
    else if ||TANG[j]-last_bit|| ≤ max_inversion then
      {continue despite this gradient inversion; do nothing for now}
    else if left_bit = j-1 then
      if TANG[j] ≥ last_bit then
        endian ← true
      else
        endian ← false
      end if
    else
      comp.append((left_bit, j-1))
      left_bit ← j
    end if
  else
    token_started = true
    left_bit ← j
  end if
  last_bit ← TANG[j]
end for
if token_started then
  comp.append((left_bit, ||TANG||-1))
end if
return comp
```

Algorithm 1: Greedy Payload *Tokenization* Using a TANG

The benefits of this greedy search approach include the following:

1. No reliance on heuristics or *a priori* knowledge of X_{ID} to produce $Comp_m(X_{ID})$
2. It allows for different endian bit ordering $\forall t_k \in X_{ID}$[2]
3. Efficient memory and computational complexity
4. High accuracy and consistency as demonstrated is Section 4.4

Calculating the TANG of a matrix of payloads X_{ID} requires $O(I \times J)$ computation and memory complexity to create the *transition matrix* and calculate the sum of each column vector x_j in that matrix. Algorithm 1 requires constant $O(n \leq 64)$[3] computation and memory complexity to execute a single loop using a TANG to output $Comp_m(X_{ID})$.

Algorithm 1 begins with a `for` loop over each index of the TANG array using the j iterator variable. Lines 6 to 9 handle the case when a particular *bit position* $x_j \in X_{ID}$ had zero transitions in the sample. This phenomenon is referred to as a *padding bit*. If the `padding` variable is set to `TRUE` to enable *padding bits* to be included in the *composition* $Comp_m(X_{ID})$, then lines 6 to 9 are effectively ignored. If `padding` is `FALSE` and *padding bits* are being excluded from $Comp_m(X_{ID})$, then these lines add the set of *bit positions* $\{x_{left\_bit}, ..., x_{j-1}\}$ being clustered—if any—to $Comp_m(X_{ID})$ and resets the other flow control variables.

Lines 10 to 19 of Algorithm 1 handle the cases when a local minimum or local maximum has already been identified. Line 11 enables gradient ascent when a local minimum was identified and the current x_j represents part of a monotonically increasing gradient of transition counts. Line 12 enables gradient descent when a

[2]Endians are introduced in Section 2.4

[3]The 64 upper limit refers to the maximum of 64-bit payloads in the standard and extended CAN frame formats. CAN frame formats are discussed in Section 2.3.

local maximum was identified and the current x_j represents part of a monotonically decreasing gradient of transition counts. If the `max_inversion` variable is set to a float value greater than 0, then line 13 allows for gradients between local minima and maxima to deviate from a strictly increasing or decreasing series of transition count values. If a local maximum or minimum was found at the previous index, then lines 14 to 17 detect whether this second *bit position* represents the beginning of an increasing or decreasing gradient of transition values. This logic is equivalent to determining the *token's* endian formatting. The final portion of lines 10 to 19 account for when a local minimum has been found next to a local maximum and a new *token* should be started. These lines record the current cluster of $x_j \in X_{ID}$ as a *token* and add it to the *composition* $Comp_m(X_{ID})$.

If no token is currently being created then lines 20 to 21 ensure a new *token* is created using the current *bit position* as its first element. Line 23 ensures that any *token* in the process of being clustered at the conclusion of the for loop is added to $Comp_m(X_{ID})$. $Comp_m(X_{ID})$ is returned as a list of tuples with each tuple representing the left most and right most *bit position* bounding each *token* $t_k \in Comp_m(X_{ID})$ on line 23.

Payload Tokenization Example.

Fig. 25 is an example payload *tokenized* using its TANG and Algorithm 1. The TANG is a $1 \times n$ row vector containing the sum each $x_j \in X_{0x42D}$ transitioned between a 0 and 1 in sequential samples $X_{0x42D,i}$. This example is taken from an eleven minute sample of *city* driving with a 2015 light duty truck. The two *time series* plots at the top of Fig. 25 represent each logically distinct *signal* present in the set of payloads X_{0x42D}. These plots represent the two distinct $t_k \in X_{0x42D}$ visible in the 64-bit TANG plot at the bottom of the figure. The vertical axis of these *time series* plots

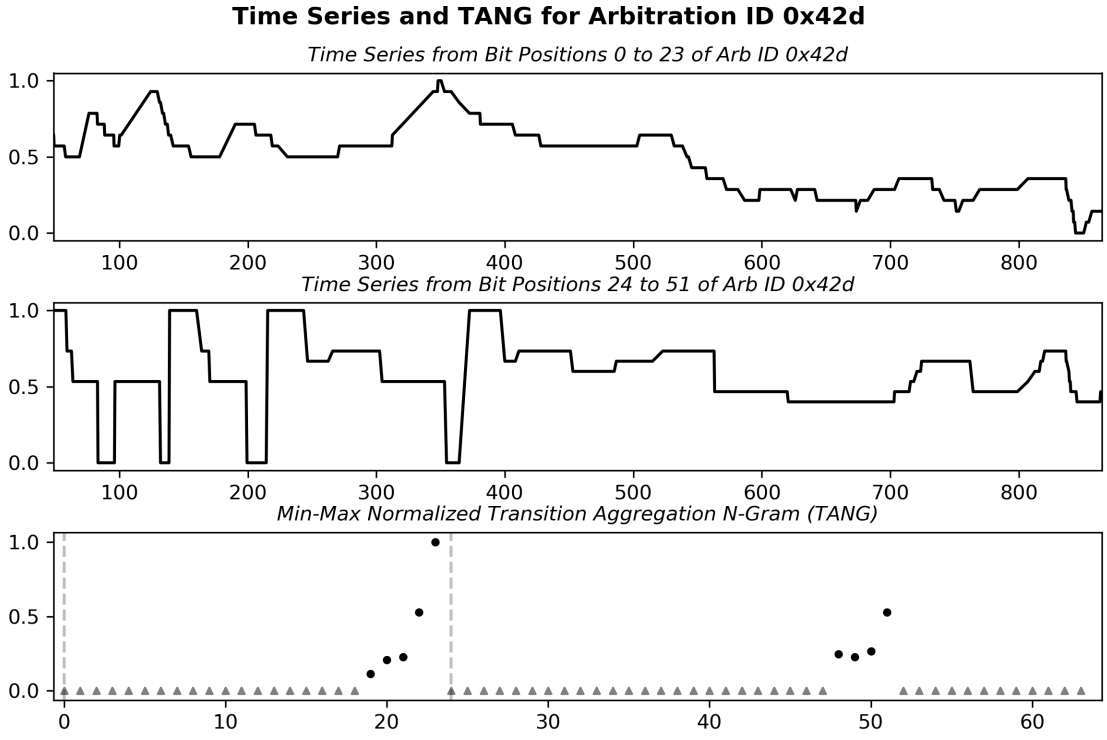


Figure 25. Tokenization Result of Payloads Using Arbitration ID 0x42D

is the unsigned integer interpretation of each *token* with magnitudes normalized to unit scale (0 to 1). The horizontal axis is the chronological index of the payloads observed in the sample. Thus, these *time series* plots may be read from left to right as the normalized unsigned numerical value the subset of *bit positions* took on as time progressed.

The third plot at the bottom of Fig. 25 is a scatter plot of the TANG for the set of payloads X_{0x42D}. The vertical axis of the TANG plot is the unit scale normalized transition count for each *bit position* in the 64-bit payloads. Higher values on this vertical axis indicate the *bit position* marked by the horizontal axis transitioned more frequently. The horizontal axis indicates the total *bit positions* in the series of observed payloads. In this example, all observed payloads associated with arbitration ID 0x42D used 64-bits.

Vertical dashed lines indicate the *most significant bit* (MSB) of a *signal*. The LSB

of each *signal* is not explicitly identified to avoid clutter. However, both the LSB and MSB are explicitly listed in the title of each *time series* plot. Grey triangles in the TANG plot indicate possible *padding bits* observed in the CAN data sample; these *bit positions* never transitioned.

Padding bits were included in $Comp_m(X_{ID})$ when producing Fig. 25. The remaining findings in this research assumed that *padding bits* were the most significant bits of $t_k \in X_{ID}$. Exploring methods for identifying when *padding bits* should or should not be used included in payload *compositions* is beyond the scope of this dissertation. However, the empirical findings in this dissertation suggest that *padding bits* should be included in payload *compositions*.

4.4 Validation

Tokenization Consistency Using Observed Payloads.

Samples from highly controlled and 'city' driving scenarios are used to evaluate the consistency of Algorithm 1. Sets of payloads X_{ID} with identical IDs and bit widths in both samples are used for this evaluation. The *Alignment Score* S metric is used to quantify the consistency of *tokenization* between samples. Accuracy cannot be evaluated due to the lack of truth data regarding the correct payload *compositions* for the vehicles studied.

The first phase of data collection described in Section 3.2 consisted of gathering driving samples from 17 different passenger vehicles during a short 'city' driving scenario. One sample was gathered from each vehicle. A k-fold cross-validation strategy with 5 folds was used to iteratively segment these samples into two subsets [11, 105]. A chronologically contiguous subset of payloads representing one fifth of the total sample was held out of the larger sample at each of the five iterations. Lexical analysis was performed on both the holdout subset and remaining sample to produce

two *token compositions*: $Comp_m(X_{ID})$ and $Comp_{m+1}(X_{ID})$. The Alignment Score S was calculated using these *compositions*. The average of these five *composition* comparisons $\overline{S}$ was recorded for each vehicle.

Table 11 lists the average Alignment Score using this k-fold cross-validation strategy for the 17 vehicles studied in the first phase of data collection. The average Alignment Score measuring *tokenization* consistency across all arbitration IDs for all 17 vehicle was 0.93. If every ID is assumed to use 64-bit payloads, this equates to an average difference of approximately four different boundary decisions between samples. This highly consistent result for a generalizable *lexical analysis* method is encouraging; however, these four dissimilar boundary decisions likely cause significant numerical interpretations of an observed data matrix X_{ID}. Additional manual adjustment by a human expert, domain specific heuristics, or tuning threshold parameters to a particular OEM or vehicle model may be necessary to further improve *tokenization* consistency. It is also the intent of this research to facilitate the rapid creating of extensive labeled data sets for training *supervised* machine learning models such as a Neural Network. An adequately trained supervised machine learning model is expected to eventually replace or augment the *unsupervised* techniques presented in this chapter. The techniques proposed in this chapter 'therefore serves as the gold standard against which newer tests can be compared. When enough data have accumulated to make [this] gold standard untenable it can perfectly reasonably be replaced by another. This can then preside until it too is toppled' [106].

Table 11. Average Alignment Score of K-Fold Cross Validation Using 'City' Driving Samples

Vehicle	**Alignment Score $\overline{S}$**	**Vehicle**	$\overline{S}$	**Vehicle**	$\overline{S}$	**Vehicle**	$\overline{S}$	**Vehicle**	$\overline{S}$
1	0.93	2	0.92	3	0.94	4	0.93	5	0.94
6	0.91	7	0.95	8	0.94	9	0.95	10	0.90
11	0.92	12	0.93	13	0.94	14	0.97	15	0.94
16	0.92	17	0.88						

The second phase of data collection involved operating vehicle 14 in various driving conditions while recording J1979 diagnostic data. The Alignment Scores comparing *tokenization* of payloads from each driving scenario are presented in Table 12. The *tokenization* strategy resulted in an average of 99.2% consistent payload boundary decisions between five iterations of the *very similar* driving scenario described in Section 3.1 using vehicle 14. Comparing the five iterations of controlled driving samples to a shorter 'city' driving sample shown in Fig. 23 led to an average of 98.7% consistent payload boundary decisions. The average Alignment Score comparing each of the five *very similar* driving scenario samples to the *dissimilar* long 'city' driving sample resulted in an average of 98.6% *tokenization* similarity. Comparing the short and long 'city' driving samples resulted in 99.3% similar *tokenization* results. Multiple iterations of the 'short' and 'long' city driving conditions were not collected.

This average Alignment Score of .99 across the three *dissimilar* driving scenarios is higher than the average Alignment Score of 0.94 measured using k-fold analysis of a single sample from the same vehicle. These results equate to an average of approximately four different boundary decisions in a 64-bit payload when measured using k-fold cross validation compared to just one differing boundary decision when comparing *dissimilar* driving samples. This difference in measured *consistency* depending on the validation strategy and driving conditions highlight the need for future work focused specifically on understanding how experimental design and validation strategy may effect findings. The goal of such work may be to propose best practices for how to employ comparison testing of unsupervised *lexical analysis* techniques when no truth data is available.

Table 12. Alignment Score of *Very Similar* and *Dissimilar* Driving Scenarios Using Vehicle 15

	Controlled (5 Iterations)	Short 'City'	Long 'City'
Controlled	0.992	0.987	0.986
Short 'City'		N/A	0.993
Long 'City			N/A

Tokenization Accuracy Using Simulated Payloads.

A combination of random walk data simulation strategies described in Section 3.3 were used to quantify the *tokenization* accuracy of Algorithm 1. The intent of this simulation is to evaluate *tokenization* accuracy in three general scenarios. The first evaluation tests payload *tokenization* accuracy using only payloads containing continuous numerical *time series* data. This is the ideal scenario given the assumptions and design of Algorithm 1. The second evaluation tests *tokenization* output when non-continuous *time series* are interwoven with continuous *time series*. The second scenario is intended to represent challenging but more realistic payloads that are representative of production vehicle CAN networks. The final scenario is simulation of payloads containing only non-continuous *time series*.

Tokenization of simulated payloads including only continuous numerical *time series* was 100% accurate. Simulated payloads composed of interleaved continuous and non-continuous *time series* resulted in *tokenization* accuracy of 96.1% on average. Algorithm 1 achieved an average of 90.6% *tokenization* accuracy with simulated payloads consisting only of non-continuous *time series*.

The *tokenization* accuracy of these simulations appear to be very good when expressed as percentages. However, a 0.906 *tokenization alignment score* for a 64-bit CAN payload means 6 boundary decisions are incorrect on average. Orphaning or incorrectly adding even a single MSB or sign bit can cause extremely different numerical

interpretations of binary data. Thus, we subjectively believe these simulation results generally agree with the empirical results from examining production vehicular CAN networks in three important ways:

1. Continuous numerical *signals* are consistently and accurately *tokenized* when adjacent to other continuous *signals*

2. When adjacent to a non-continuous *signal*, *off by one* errors in identifying the MSB of a continuous *signal* may occur; this causes potentially large interpretation errors

3. While apparently capable of identifying the general position of a non-continuous *signal* in a payload, consistent *off by one* errors in identifying the MSB and LSB of those *signals* means their numerical interpretation should be assumed to be unreliable.

4.5 Conclusions

The empirical and simulated findings suggest that the unsupervised *lexical analysis* methods proposed in this chapter may serve as a viable 'gold standard' for CAN payload *tokenization*. However—like 'gold standard' tests in the medical field—our hope is that this may simply serve 'as the gold standard against which newer tests can be compared. When enough data have accumulated to make [this] gold standard untenable it can perfectly reasonably be replaced by another' [106]. Specifically, the *time series* extraction provided by Algorithm 1 combined with the use of J1979 diagnostics for truth data and manual labeling should enable the production of reasonably large labeled data sets from proprietary passenger vehicle networks. These labeled data sets could then be used to train supervised classifiers or enable development and testing of even more accurate payload *lexical analysis* techniques.

V. Unsupervised Semantic Analysis of Automobile Time Series Data

This chapter introduces two techniques intended to achieve unsupervised *semantic analysis* of *signals* identified by *lexical analysis* of observed CAN network data. These are intended to be functionally similar to the semi-supervised *label propagation* technique proposed by Glennan et al. discussed in Section 2.5. In this context, unsupervised clustering enables *signals* to be grouped into semantically homogeneous groups. Some of those groups may then be labeled using limited truth data such as J1979 diagnostic information and manual reverse engineering[1]. These clusters also serve as a potentially useful feature for Intrusion Detection Systems (IDS) to detect when one or more *signals* deviate from the majority of *signals* in an otherwise highly correlated cluster.

Section 2.5 highlighted the significant differences between analyzing the CAN protocol and the text-based protocols being studied by Glennan et al. The expectation that CPS in automotive CAN networks exclusively produce univariate *time series* data is one of the most important differences. The proposal by Glennan et al. is intended to account for a heterogeneous population of protocols and data types such as numbers, text, and flow control meta-data. Meta-data such as checksums or categorical variables like transmission gear are expected to be present in automotive CAN networks; however, empirical findings in this dissertation and related research demonstrate that the vast majority of data is *time series* generated by CPS throughout the vehicle [21, 22, 107, 108, 109, 110]. This enables *semantic analysis* to focus on *time series* analysis and clustering.

This chapter achieves unsupervised *semantic analysis* with a completely unlabeled data set in two phases. Subset selection of continuous numerical *signals* forms the

[1]The J1979 diagnostic standard was introduced in Section 2.3

first phase. The intent of this subset selection is to differentiate *signals* expected to be successfully *tokenized* from other data such as checksums or non-continuous *signals*. The second phase attempts to cluster continuous numerical *signals* to find highly correlated subgroups among the observations.

Glennan et al. used the k-Means clustering algorithm to achieve their particular *semantic analysis* implementation [9]. The k-Means algorithm requires the number of clusters to be specified *a priori* [11, 105]. This is not an acceptable requirement in the context of analyzing vehicle networks which may each have a different number of semantically unique types of *signals* which an independent researcher is unlikely to know *a priori*. Density-Based Spatial Clustering (DBSCAN), k-Nearest Neighbors (k-NN), and Agglomerative Hierarchical Clustering (AHC) are alternative clustering methods used by other reverse engineering proposals. Those proposals and their use of clustering are discussed in Appendix A.

DBSCAN is not an ideal clustering technique in the context of unsupervised reverse engineering (RE) because it may exclude *outliers* in a data set. k-NN, like k-Means used by Glennan et al., requires the number of clusters to be known *a priori*. In each case, there is a requirement for the researcher to already have some insight about the vehicular network or be willing to throw out data. Several other techniques used by related work were considered to achieve unsupervised *semantic analysis* including Principal Component Analysis (PCA) [11, 15, 28, 45, 46, 105, 111]. Ultimately, agglomerative hierarchical clustering (AHC) proved to be a viable clustering strategy. This is because it only requires specifying a maximum *distance* threshold and linkage strategy for measuring distance between clusters [11, 105]. AHC was introduced in Section 2.6. Selecting these features does not require extensive *a priori* information about a particular vehicle. Thus, AHC is uniquely viable in the context of unsupervised *semantic analysis* of *signals* extracted from proprietary CAN networks.

Clustering strategies were tested using various statistical features of *signals* such as mean and standard deviation. The large and unpredictable variability between driving samples revealed that descriptive statistics like mean and standard deviation were unreliable features to use for clustering similar *signals*. This is due to their inconsistent relative magnitudes across *dissimilar* driving samples and network configurations.

Pairwise descriptive statistics such as the Pearson's Correlation Coefficient (PCC, Pearson's ρ) proved to be highly reliable for quantifying semantically similar *time series*. The bioinformatics community routinely publishes papers focused on clustering similar genes using pairwise statistics such as mutual information, cross entropy, and PCC [111, 112]. Clustering results using pairwise statistics are referred to as 'Relevance Networks' or 'Correlation Networks' in that domain. One example of such research is a proposal by Butte and Kohane which used a combination of pairwise mutual information and thresholding [112]. Their proposed method for generating clusters ('Relevance Networks') was to manually adjust a threshold parameter and the subset of data being clustered until a subjectively interesting connected graph structure was produced.

A proposal by Fukushima et al. also used pairwise correlation coefficients with the DPClus clustering algorithm (a density based method like DBSCAN) and several domain specific algorithms. Their proposal produced a graph structure partitioned into sub-graphs which represented clusters of correlated genes [111]. Their work highlights the prevalence of proposals based upon hierarchical clustering methods using pairwise statistics.

It is important to note that the problem of finding a graph partition which maximizes the correlation of *signals* in each partition while minimizing the correlation between partitions may be similar to the Sum-Max Graph Partitioning Problem

[113, 114]. Watrigant et al. published a pair of papers which demonstrated that the problem is NP-hard and a greedy or heuristic solution is a viable but sub-optimal solution [113, 114]. Another area of similar research is community detection in networks. A 2016 survey of community detection research by Fortunato and Hric highlight the lack of validation methods and performance metrics in the domain. It also highlights the fact that community detection in networks remains an open problem with a very active research community [115]. Based on similar work being done in bioinformatics, it seems likely some form of weighted graph clustering may be a viable approach for performing *semantic analysis* of automotive *time series*. The correlated clusters produced by techniques in this chapter coupled with causal links discussed in Chapter VI may provide the features necessary to enable community detection research in the CPS network domain.

5.1 Assumptions and Limitations

Vehicle's are assumed to include several CPS monitoring highly correlated physical processes. An example of this phenomenon is the likely correlation between the output of CPS monitoring the vehicle's two or more wheel rotations per minute (RPM) and those reporting the vehicle speed. There is an approximately linear relationship between wheel RPM and current vehicle speed. Because even motorcycles have at least two wheels, it is reasonable to assume there is at least two highly correlated *signals* on the CAN network.

Signals for different physical processes being monitored are expected to be weakly correlated compared to *signals* related to the same process. This assumption is the purpose for selecting the single linkage cluster dissimilarity measurement strategy. Single linkage evaluates two clusters of *signals* to be nearby so long as at least one *signal* in cluster A is nearby to another in cluster B. Thus, when looking at the

dendrogram a series of cluster fusions are expected to occur at relatively small inter-cluster dissimilarity levels when *signals* from the same physical process are being combined into one cluster. Then, a distinct vertical gap occurs in the tree reflecting the relatively significant dissimilarity between all *signals* related to different physical processes.

Another assumption used to perform subset selection is that *signals* produced by CPS monitoring analog physical processes like wheel RPM, steering wheel angle, or brake pressure are more *diverse* than other *signals* present on the network. Diversity is measured using the Shannon Diversity Index statistic presented in the next section [116]. Phrased another way, this assumption means *signals* with few unique values or heavily favor only a few values are expected to not be continuous numerical *signals* that were correctly *tokenized* during *lexical analysis*. Those *signals* are assumed to represent subsets of *bit positions* $x_j \in X_{ID}$ which require additional *lexical analysis* before being considered for *semantic analysis*.

5.2 Subset Selection of Continuous Numerical Time Series

The Shannon Diversity Index proved to be a reliable statistic for selecting a subset of continuous numerical *signals* from data produced by networks representing several Original Equipment Manufacturers (OEMs), model years, and vehicle models. It measures the number of unique members in a population and the proportion of the total population each represents [48]. Equation 7 is a method for calculating this statistic given a population of decimal numbers.

$$H = -\sum_{i=1}^{n} x * log_2(x) \tag{7}$$

The variable x represents the proportion of payloads in a *signal* which have the same numerical interpretation. For example, consider a *signal* composed of

100 observed payloads. A particular numerical interpretation is observed 10 times in the total of 100 observations. Then that particular part of the sum would be $(10/100) \times log_2(10/100) \approx -0.332$. The upper limit n represents the total number of unique numerical values present in the population of payloads the *signal* is composed from. H is the Shannon Diversity Index. The base of the logarithm used affects the units which H is expressed. H is expressed in *bits* when using log_2, *decimal digits* with log_{10}, and *natural units* with log_e; a *decimal digit* is about $3\frac{1}{3}$ *bits* [116]. The range of H when using log_2 is 0 to the *bit width* of the *signal*. The minimum occurs when a *signal* is only one value repeated in every observation. The maximum Shannon Index occurs when two conditions are met. First, the number of unique values n observed in the *signal* but be 2 to the power of the *signal's bit width*: $2^{n_{bit\ width}}$. Second, each unique value x_i must represent an equal proportion of the total observations. Expressed in another way, the *signal* must be composed of $2^{n_{bit\ width}}$ unique numerical values and each value must be observed the same number of times.

The Shannon Index is used in this context with the assumption that *signals* describing continuous numerical processes are more diverse than *signals* related to categorical data like turn signal activation. Selecting a subset of only continuous numerical *signals* can be achieved by sorting the total population of *signals* produced during *tokenization* by Shannon Index. A tunable percentage parameter t_{subset} is used to select a percentage of the total population with the largest Shannon Index values. Increasing this percentage parameter increases the quantity of *signals* that are labeled as continuous numerical *signals* in the *signal* clustering phase presented in Section 5.3. However, this also increases the possibility that *signals* which are not actually continuous numerical *signals* become mislabeled. As with all unsupervised clustering methods, there is no 'magic' setting for a parameter which is optimal for all data sets.

Fig. 26 provides a graphical guide for interpreting the notched box and whisker

plots of Shannon Index values for studied vehicles shown in Fig. 27. The critical piece of information is the Interquartile Range (IQR) expressed by the box in each plot. The box represents 50% of observed Shannon Index values for *signals* produced by the vehicle and extracted during *lexical analysis*. Each whisker is either $1.5 \times IQR$ from the nearest edge of the box or the *minimum* or *maximum* observed Shannon Index. Observations that exceed either whisker are considered *potential outliers* [117].

Fig. 27 may help provide insight into how to best leverage the Shannon Index statistic to perform subset selection. This chapter focuses on using a population percentage threshold to select a subset of *signals* generated during *lexical analysis* which represent the most diverse *signals*. An alternative approach may be to perform subset selection by excluding *signals* which do not meet a minimum Shannon Index threshold. Fig. 27 shows approximately what percentage of the total population of *signals* extracted from each vehicle would be excluded for different threshold values. The percentage based approach proposed in this section ensures a relatively consistent quantity of *signals* are included during *semantic analysis*. The alternative minimum threshold approach may cause the majority of *signals* from one vehicle to be included in the subset while excluding the majority in another vehicle. Comparing how subset selection would be effected by using a minimum Shannon Index threshold of three with the *signal* populations from vehicles five and eleven demonstrates how such a scenario might occur in practice.

It is not immediately clear whether a percentage based, 'fixed', a hybrid threshold approach for Shannon Index based *signal* subset selection technique is superior. *Lexical Analysis* of vehicle five produced a population of *signals* with approximately 75% containing less than two bits of information. It is unlikely that a continuous numerical *time series* can be expressed using only two bits. Thus, a percentage based subset selection method is unlikely to effectively filter non-continuous *time series*.

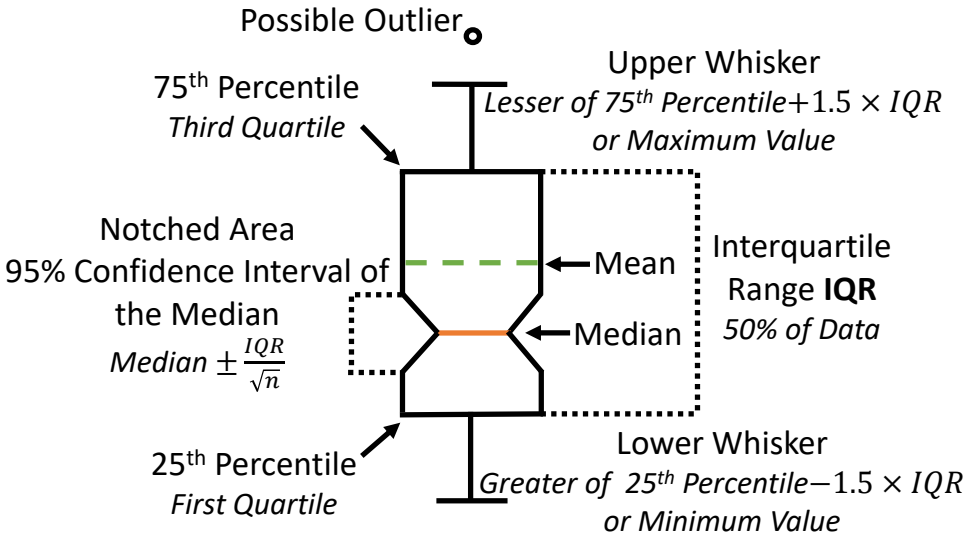


Figure 26. Features of a Notched Box Plot

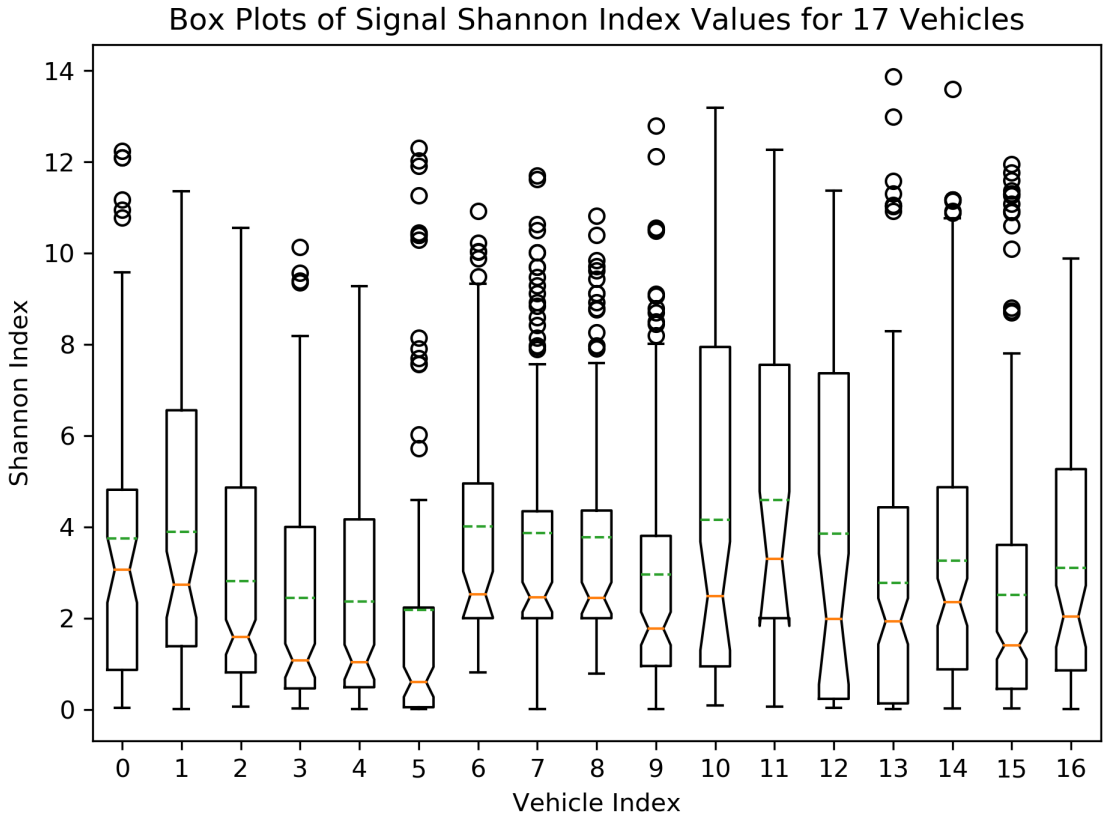


Figure 27. Notched Box Plot of Observed Signal Shannon Index Values by Vehicle

The varied distribution of Shannon Index values across the 17 vehicles studied also highlight that it is also unlikely that a 'fixed' minimum threshold is effective for all vehicles; some sort of adjustment may be necessary based on the vehicle OEM or make. Such adjustments are expected to require domain specific heuristics or *a priori* knowledge which this research is explicitly avoiding in order to remain as generalizable as possible. Thus, the percentage based approach is used in this chapter based upon the assumption that it is a more generalizable thresholding strategy.

5.3 Time Series Clustering Using the Pearson's Correlation Coefficient

The *signal* clustering process begins with generating a correlation matrix C using the Pearson Correlation Coefficient (PCC). PCC is a measure of linear correlation between two variables [118]. In this case, the unsigned integer interpretation of each pair of *signals* produced during *tokenization* are being correlated. The `DataFrame.corr()` method in Python's Pandas library generated the correlation matrix used to produce the results presented in this chapter [102].

Two modifications are made to the correlation matrix C to use it as a *dissimilarity* matrix for Agglomerative Hierarchical Clustering (AHC). First, any negative values in C are set to 0. Negative PCC values indicate two *signals* were negatively correlated in the sampled data. The current intent of this proposal is to only consider *positively* correlated *signals* extracted during *lexical analysis*. Consideration of how to best leverage information about negatively correlated *signals* is reserved for future work. The second modification is to perform the matrix subtraction $1 - C$. This is done to convert large PCC results (near 1) between *highly correlated signals* to *small dissimilarity* values. Likewise, uncorrelated *signals* with a PCC near 0 now have *large* dissimilarity values near 1. This modified correlation matrix C is referred as the distance matrix D.

Single linkage AHC using the distance matrix D was performed using the Python SciPy library's `hierarchy` module in the clustering package `scipy.cluster` [119]. Dendrograms shown in this chapter and Appendix B were also produced using this module and the Python Matplotlib library [104].

This clustering strategy is implicitly targeted for CAN networks with few continuous numerical *signals* with unique behavior. This is because clusters are formed using the pairwise PCC statistic. If a continuous numerical *signal* corresponds to a unique physical process then it is unlikely that another *signal* in the network is highly correlated. By requiring every *signal* to be correlated with at least one other *signal* to be clustered and labeled, an implicit false positive filtering process is performed. Specifically, it is possible that edge cases or an overly inclusive subset selection `threshold` parameter choice results in one or more improperly *tokenized signals* being included in the subset of continuous numerical *signals*. By requiring these improperly *tokenized signals* be correlated to another *signal* in the subset, this effectively ensures these false positive subset selection errors require another highly correlated *tokenization* error to occur before both false positives are incorrectly clustered.

Table 13. Summary Results of Sample Sizes, Tokenization, and Signal Clustering

	Make	Model	Model Year	Total Payloads	Total Arb IDs	Total Static IDs	Total Short IDs	Avg. Payload Bit Width	Signals Extracted	Avg. Signal Bit Width	Avg. Signal Shannon Index	Total Singleton Clusters	Total Non-Singleton Clusters	Avg. Cluster Population	Total Clusters in Subset	Avg. Subset Cluster Population
0	1	1	2009	626,373	29	4	0	50.2	80	15.6	3.75	23	11	4.64	10	4.90
1	1	2	2011	300,385	63	18	0	49.5	142	15.6	3.89	46	18	4.39	8	6.62
2	2	3	2014	247,850	122	47	6	58.2	304	14.3	2.81	100	35	5.06	20	5.90
3	3	4	2015	336,007	81	19	3	64.0	258	15.4	2.45	86	31	4.45	17	5.94
4	3	4	2017	284,194	85	20	2	64.0	261	15.9	2.37	78	29	5.14	18	5.67
5	3	5	2010	483,652	26	3	0	64.0	121	12.2	2.18	27	18	4.50	13	2.77
6	4	6	2012	248,940	29	0	0	53.8	87	17.9	4.02	46	9	4.33	4	4.50
7	4	6	2015	349,732	42	0	0	51.1	127	16.9	3.86	89	12	3.08	6	3.50
8	4	7	2012	279,407	42	0	0	51.6	117	18.5	3.77	67	15	3.27	7	3.86
9	5	8	2015	437,282	45	4	0	57.2	166	14.3	2.96	59	20	4.65	14	4.93
10	5	9	2010	367,813	30	4	2	51.5	87	16.8	4.16	34	15	3.27	12	2.92
11	6	10	2009	486,394	20	6	0	46.8	44	17.1	4.59	22	5	2.60	4	2.75
12	7	11	2009	323,327	38	13	0	55.8	72	18.7	3.85	32	10	2.70	6	3.17
13	7	12	2008	1,129,028	88	31	0	55.9	206	15.9	2.78	76	17	4.47	7	7.14
14	7	13	2012	266,611	88	38	0	55.9	172	16.4	3.26	61	20	4.25	11	5.09
15	7	14	2017	414,283	124	51	0	58.6	293	14.2	2.51	81	36	5.06	23	6.43
16	8	15	2009	273,481	27	0	0	64.0	110	15.7	3.10	56	14	3.50	9	3.56
				Average	57.6	15.2	0.8	56.0	155.7	16.0	3.3	57.8	18.5	4.1	11.1	4.7
				Standard Deviation	34.0	17.0	1.6	5.6	81.3	1.7	0.7	24.8	9.2	0.8	5.7	1.4

5.4 Validation

Table 13 provides a summary of the unsupervised *lexical* and *semantic analysis* pipeline discussed in Chapters IV and V. These data were collected having owners drive according to a scripted scenario and route. The driving script took approximately 10 minutes to complete and took place in 'city' driving conditions. We propose that the large standard deviation with respect to the average for each feature listed in Table 13 indicates that there is significant variability between passenger vehicle CAN network configurations. Additionally, the highly similar payload bit width and *signal* bit width apparently used by Original Equipment Manufacturer (OEM) two suggest there may be a degree of predictability about network configuration if the OEM is known *a priori* or can be inferred from the data.

Example output of the semantic analysis techniques described in this chapter are presented in Fig. 28, 29, and 30. Section 4.3 described how to interpret these *time series* plots; the cluster numbers listed refer to the arbitrary order the cluster's branch decision was made according to the dendrogram. Each of these examples were produced by different OEMs, model types, and model years in the population studied. Similar results were found across all 17 vehicles tested. *Signals* in the clusters shown in Fig. 28 and 29 have been removed to reduce figure height. These *signals* clusters demonstrate how passenger vehicle networks often generate highly correlated *signals* across a range of arbitration IDs. These similarities are consistently captured by the *semantic analysis* approach proposed in this chapter. Unfortunately, J1979 diagnostic information could not be collected during the first phase of data collection. Therefore, it is difficult to objectively assess whether one or more of these clusters corresponded to a particular physical process.

Fig. 31 is an example cluster produced by vehicle 14 when J1979 diagnostic information was collected during the sample. The vehicle responded to the diagnostic

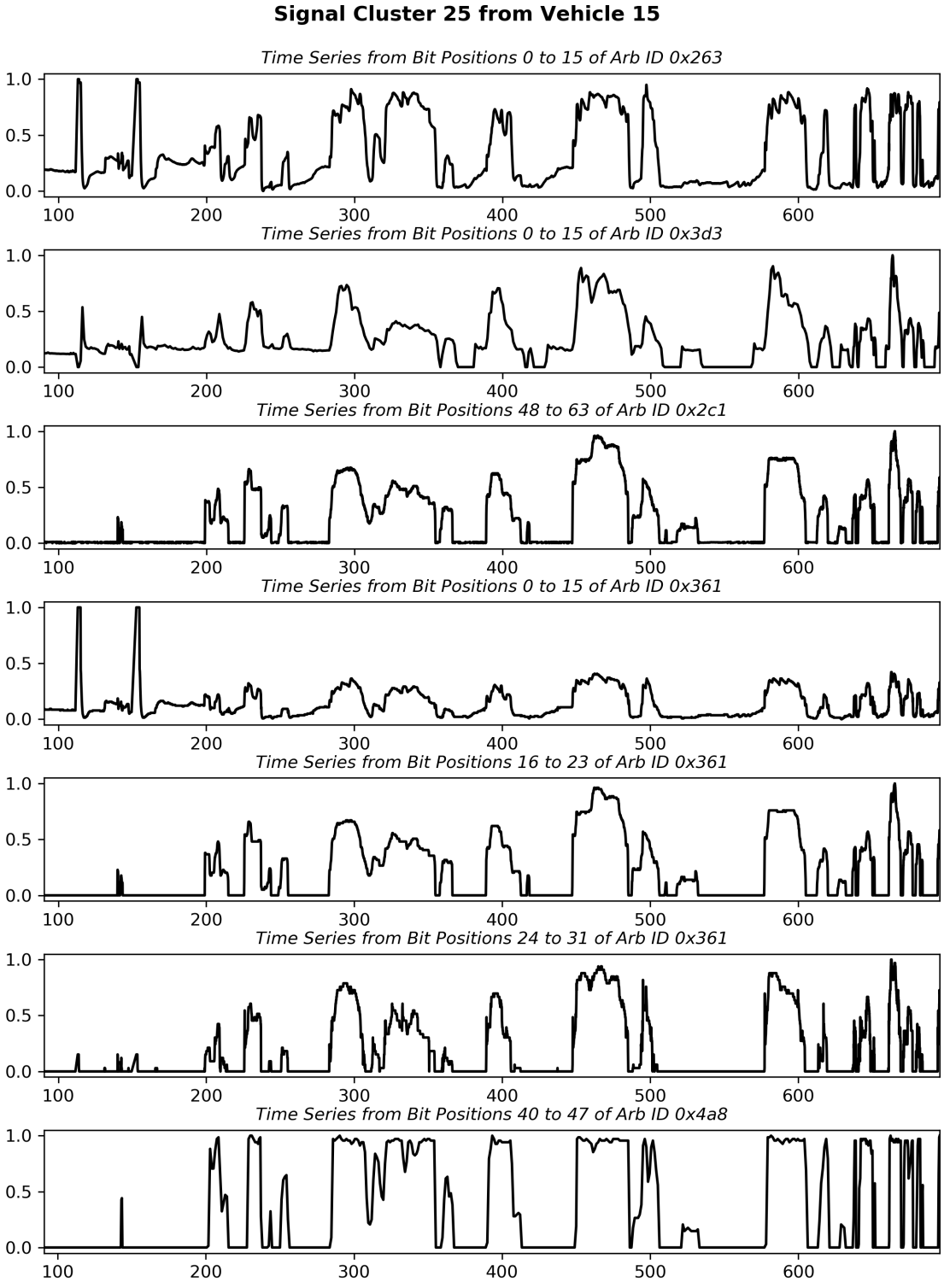


Figure 28. Signal Cluster Result from Vehicle 15

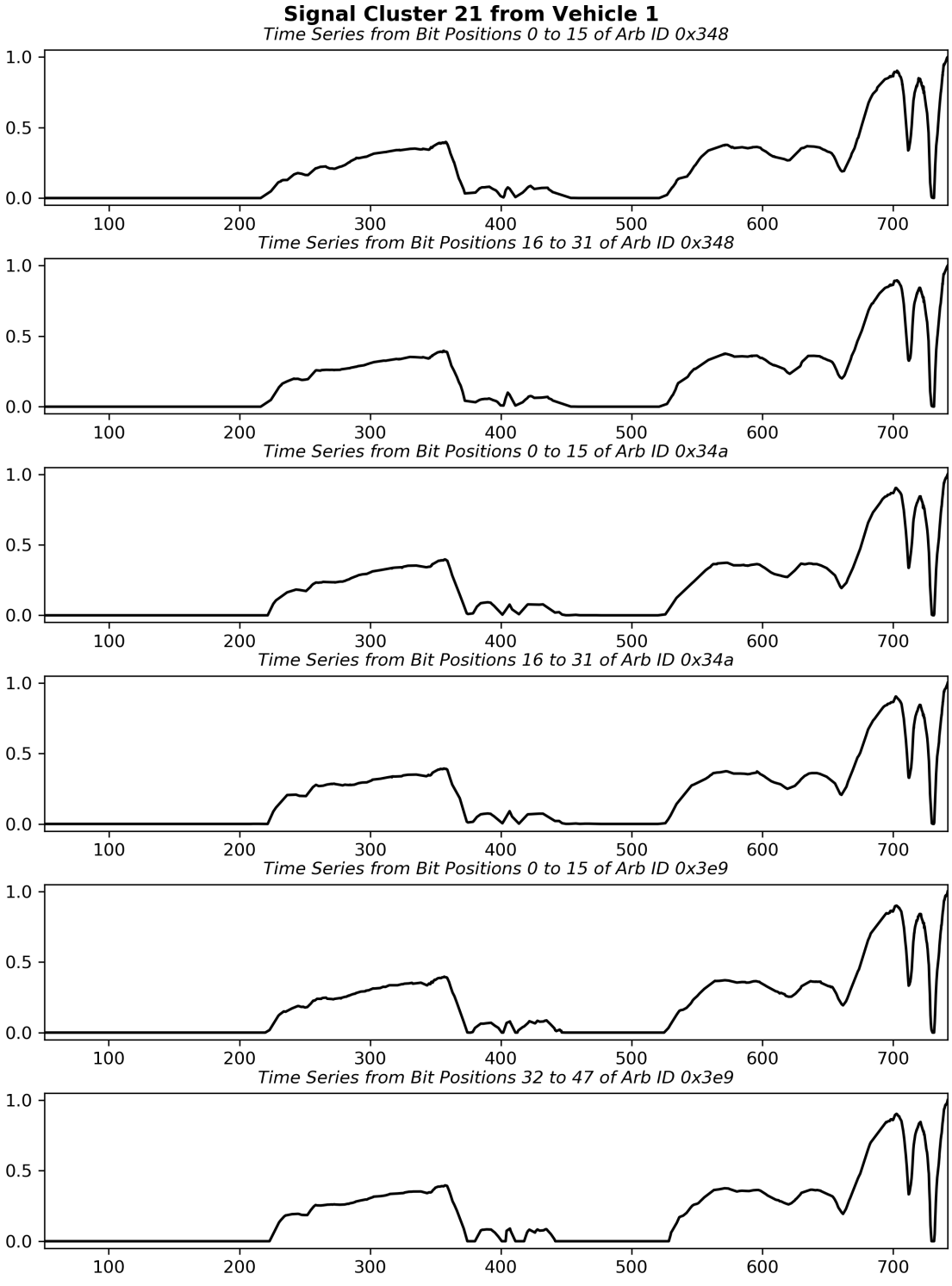


Figure 29. Signal Cluster Result from Vehicle 1

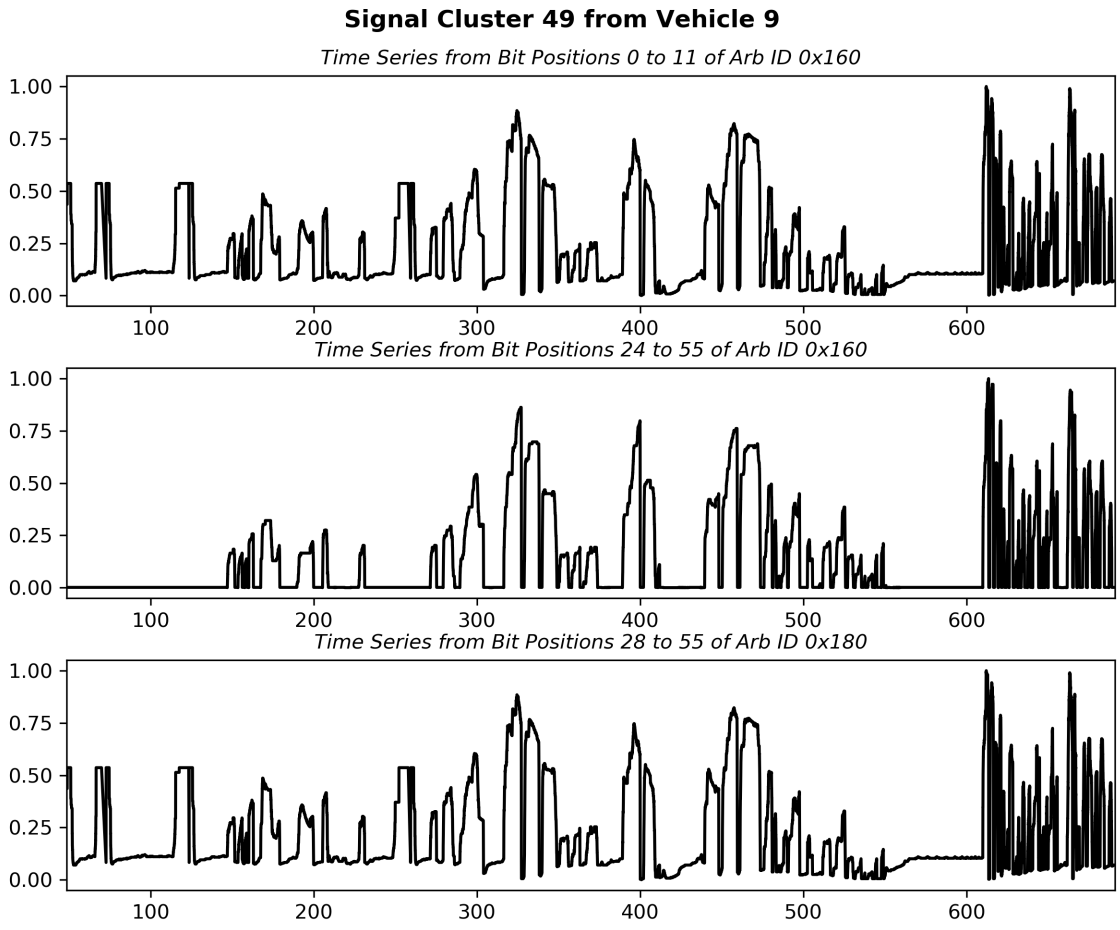


Figure 30. Signal Cluster Result from Vehicle 9

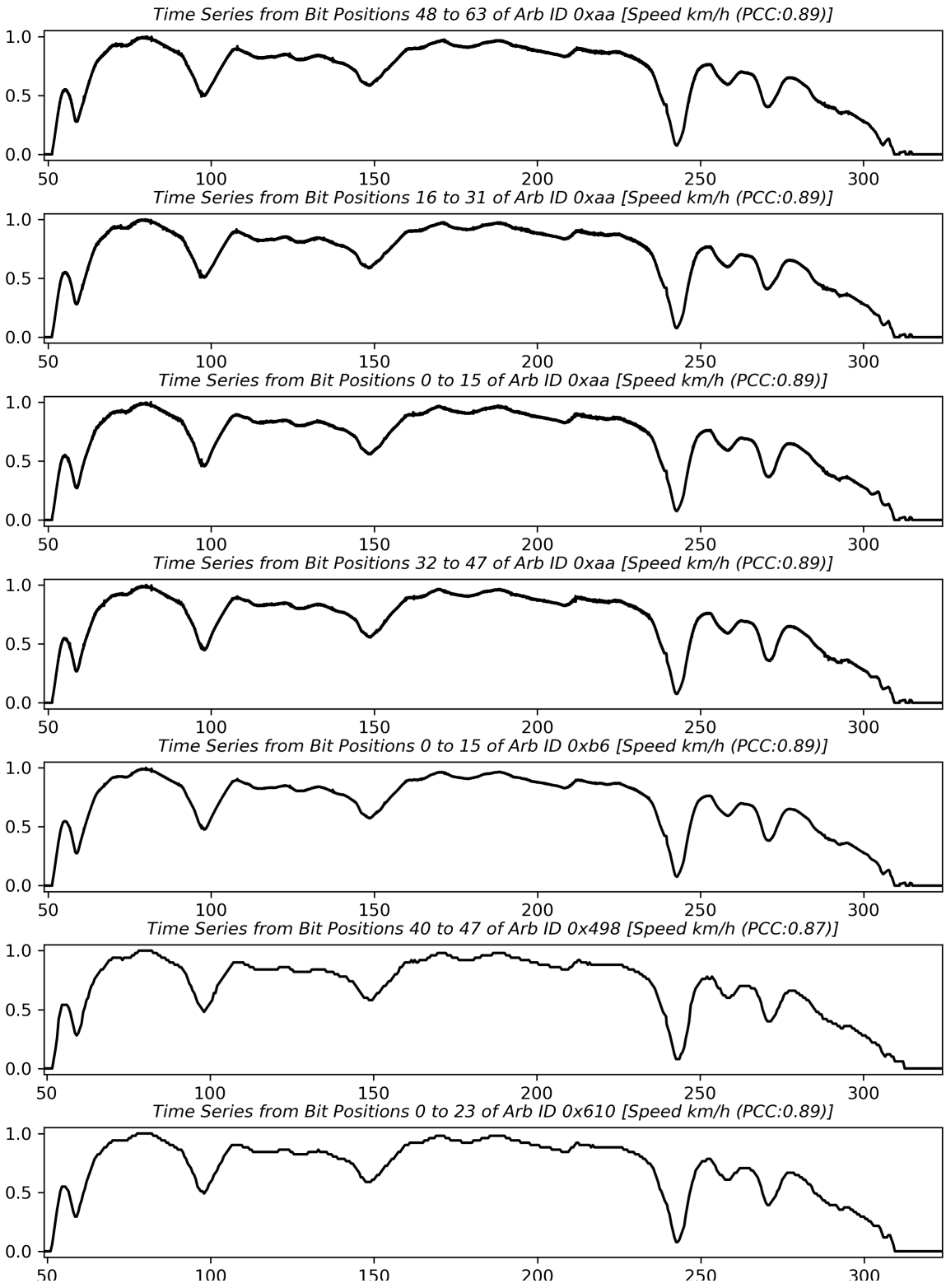


Figure 31. Signal Cluster Result from Vehicle 14 With Signals Labeled Using Correlation With J1979 Diagnostic Responses

requests listed below; their value over time is presented in Fig. 32. Each *signal* subplot lists the PCC for the most correlated J1979 service—if any. This particular example demonstrates that AHC identified a homogeneous cluster of vehicle speed *signals*. The PCC of only 0.89 on average would likely be 1.00 if more than 7 responses were received during this approximately 5 minute 'city' driving sample. Informal discussions with professionals in the automotive industry suggest that this infrequent response rate is typical when requesting multiple diagnostic services while driving passenger vehicles.

- PID 0xC: Engine RPM (7 responses)
- PID 0xD: Speed km/h (7 responses)
- PID 0x11: Throttle % (6 responses)

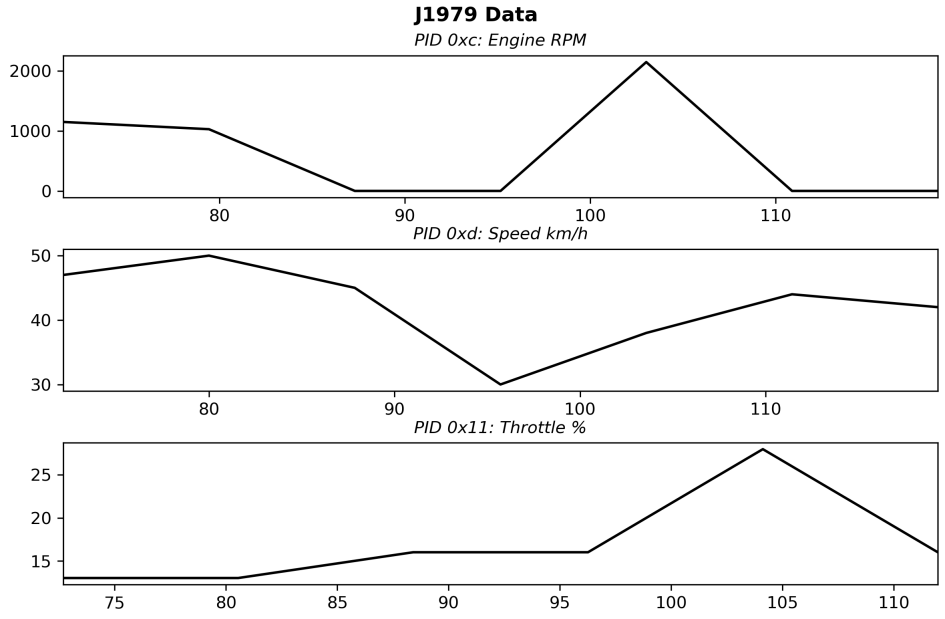


Figure 32. J1979 Diagnostic Service Responses for 'City' Driving Sample

5.5 Conclusions

The empirical findings presented in Section 5.4 indicate that the *semantic analysis* approach proposed in this chapter is consistently effective across a range vehicles and driving conditions. However, "it can be hard to assess the results obtained from unsupervised learning methods, since there is no universally accepted mechanism for performing cross-validation or validating results on an independent data set... there is no way to check out work because we do not know the true answer–the problem is unsupervised" [105]. *Signal* labeling using J1979 diagnostic data and cluster consistency using the Jaccard Index may enable some degree of quantitative validation for the *semantic analysis* strategy proposed in this chapter. However, access to DBC files provided by vehicle OEMs would enable much more comprehensive and definitive findings since those files include the correct semantic label for each *signal* in the vehicle.

VI. Empirical Data Modeling Using Automobile Time Series Data

This chapter adheres to the steps and terminology prescribed in the manual written by Ye et al. (co-authored by Dr. Sugihara) to use EDM methods with *signals* for vehicle speed, engine RPM, and brake pressure collected from vehicle 14. The pairwise relationships between vehicle speed, engine RPM, and brake were selected to demonstrate the viability of EDM in the context of automated reverse engineering of CAN networks for two reasons. First, SAE J1979 diagnostic data is available to confirm that some *signals* produced during *tokenization* are in fact vehicle speed or engine RPM [20]. Second, there is an intuitive understanding that causal relationships exist between these features such as increased engine RPM affecting vehicle speed after a brief time delay. This allows for a degree of subjective validation about whether the results are *reasonable* in the context of automotive network analysis.

Objective validation is achieved using comparison testing of EDM results based on samples collected from a single vehicle operated according to the controlled and short 'city' driving scenarios. The relative behavior of these features in either scenario are shown in Fig. 33 and 34. In both cases, there is a fairly clear visual relationship between an increase in engine RPM causing an increase in vehicle speed after a brief delay. Likewise, increased brake pressure consistently causes a noticeable decrease in vehicle speed.

Vehicle speed, brake pressure, and engine RPM *signals* were extracted using the *tokenization* process described in Chapter IV. The vehicle speed and engine RPM *signals* were confirmed to be such using post-hoc correlation with SAE J1979 diagnostic data [20]. The brake pressure *signal* was identified using data collected in the controlled driving scenario described in Section 3.2 and shown in Fig. 33. Because the brake was only applied at the beginning, mid point, and end of the driving sample,

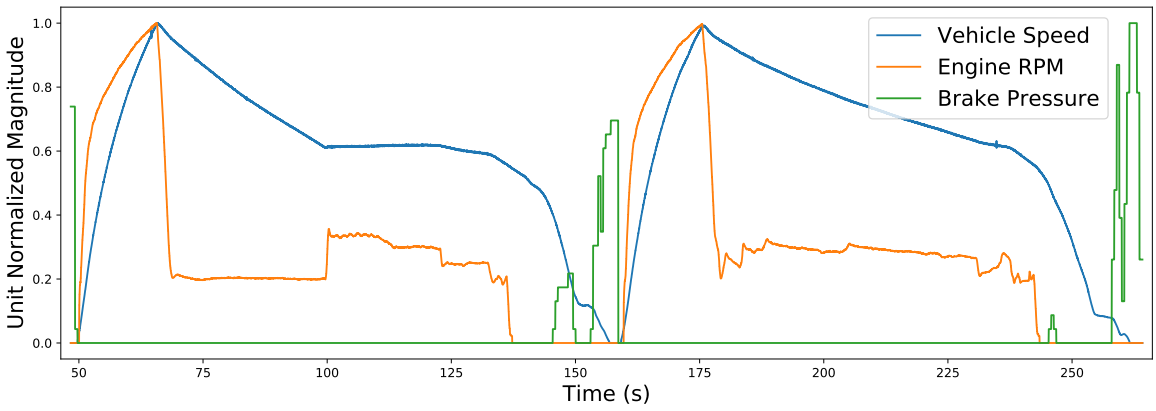


Figure 33. Example of Brake Position, Engine RPM, and Vehicle Speed Time Series from a 3 Minute Controlled Driving Scenario

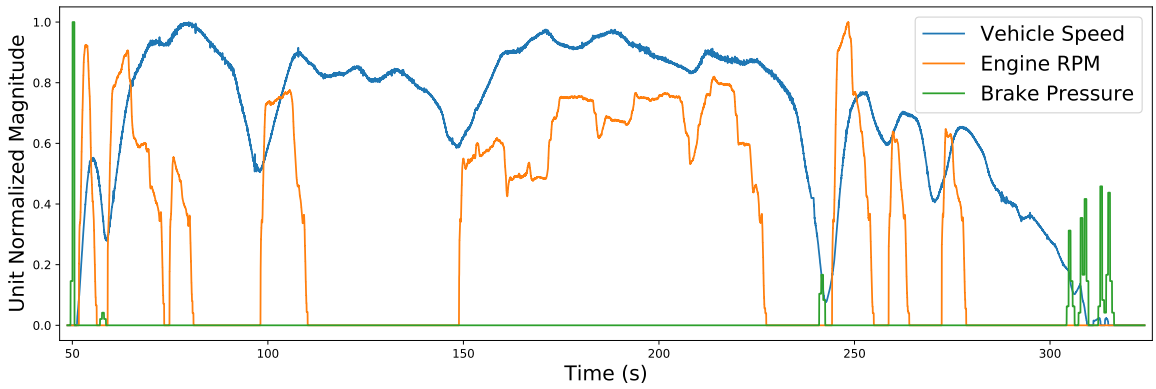


Figure 34. Example of Brake Position, Engine RPM, and Vehicle Speed Time Series from a 5 Minute 'City' Driving Scenario

manual reverse engineering quickly identified a *signal* which matched that behavior using the output of *lexical analysis*. The subsets of *bit positions* for these specific engine RPM, vehicle speed, and brake pressure *signals* were identically *tokenized* for both driving conditions studied.

Using Phase Portraits to Understand System Behavior.

Before delving into the EDM techniques introduced in Section 2.7, it may be helpful to study the *phase portraits* of the vehicle speed, engine RPM, and brake pressure *time series* being used. The *Simplex Projection* primer hosted at the University of

California San Diego's Sugihara Lab web site suggests using *phase portraits* as a precursor to EDM analysis to gain an intuitive sense for the behavior of the system being studied [17]. The primer presents the concept of *phase portraits*—or, 'lagged-coordinate embedding'—as a two or three dimension plot of a *time series* against lagged versions of itself. These lagged dimensions may be thought of as the value of the *time series* at time t, $t + \tau$, and $t + 2\tau$ with τ being the magnitude of time between each dimension. Fig. 35, 36, and 37 show the *phase portrait* of each feature being studied using *time series* from the short 'city' driving scenario. A magnitude of one was used for the lag τ.

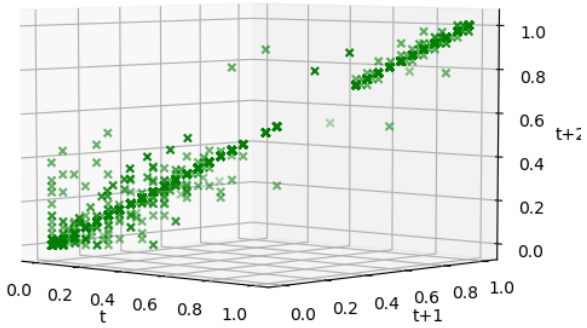


Figure 35. 3-Dimensional Plot of Brake Pressure Time Series Relative to Two Lags of the Same Time Series ('City' Driving Scenario)

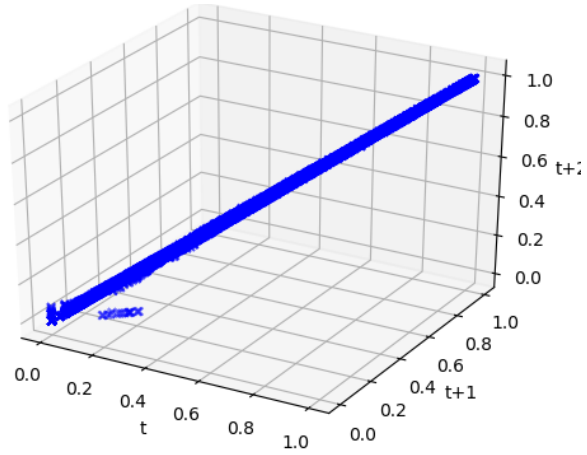


Figure 36. 3-Dimensional Plot of Engine RPM Time Series Relative to Two Lags of the Same Time Series ('City' Driving Scenario)

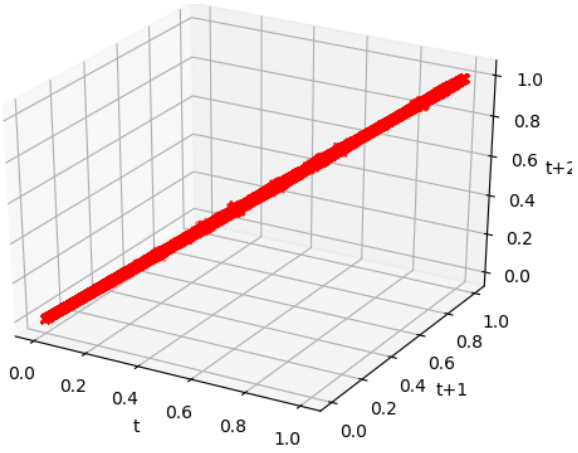


Figure 37. 3-Dimensional Plot of Vehicle Speed Time Series Relative to Two Lags of the Same Time Series ('City' Driving Scenario)

Phase portraits produced using time series which exhibit deterministic chaos are expected to have one or more visually perceptible trends. Systems with globally linear behavior, "that is tomorrow's value is proportional to today's value," produces a *phase portrait* with a strong trend along a single vector [17]. Globally linear behavior is visibly evident in the *phase portraits* of vehicle speed and engine RPM. This makes sense since the sample frequency for these *time series* is in the millisecond range and these physical features necessarily change in relatively small increments at that time scale. Conversely, the brake pressure *phase portrait* includes evidence of more dynamic behavior. This too makes sense considering that a driver *can* significantly change the pedal position within a few milliseconds. This increased potential for dynamic behavior is reflected by the cloud of points around the otherwise linear vector created from moderate changes to braking pressure input over time.

The pronounced linearity of the vehicle speed and engine RPM *phase portraits* suggests that statistical modeling methods such as auto-regressive linear models and Granger Causality may be viable alternatives to EDM [1, 17, 88]. However, these alternatives may not fully capture the dynamic behavior of braking input and other vehicle processes. The ability to work with both globally linear and dynamic systems is the motivation for using EDM technique instead of these alternatives.

Sections 6.2 and 6.3 use Simplex Projection and S-map to quantify the linearity observed in the *time series* of the three features studied. Simplex Projection is also used to make an educated guess for the optimal embedding dimension to use during *shadow manifold* reconstruction using these *time series*[1]. Multi-view embedding and Convergent Cross Mapping (CCM)—two other EDM techniques—are later used in Sections 6.4 and 6.5 to demonstrate that these globally linear features can still be used to create a highly accurate predictive model of dynamic features like braking pressure. Such models may significantly improve the accuracy of Intrusion Detection Systems (IDS) of passenger vehicles. Likewise, causal relationships identified using CCM could potentially be used to significantly expand the scope of the *label propagation* concept discussed in Chapter V.

6.1 Assumptions and Limitations

One limitation of EDM analysis is that behavior that is idiosyncratic to a particular driving scenario is reflected in the results. For example, the highly controlled driving scenario involved the driver only applying braking pressure within a relatively small range of vehicle speeds. A driver can obviously apply braking pressure at any vehicle speed; however, data collected during this particular driving scenario appears to have caused EDM results to reflect a potentially artificially strong causal relationship from vehicle speed onto braking pressure. Analysis of the *dissimilar* and relatively unscripted 'city' driving scenario is intended to highlight this limitation of EDM analysis of samples that are know to include atypical behavior. There may be other subtle idiosyncrasies of the driver, vehicle, or 'city' driving scenario which caused artificially strong or weak causal relationships to be found using EDM methods. We assume the impact of such sample specific influences are minimal for the

[1]Section 2.7 introduced *shadow manifold* reconstruction, manifold embedding dimensionality, and other EDM concepts discussed in Chapter VI

purposes of demonstrating that EDM is a viable unsupervised automotive network *semantic analysis* method.

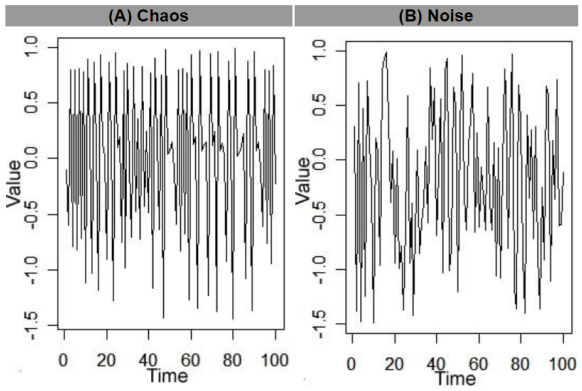


Figure 38. Examples of Time Series Representing (A) Deterministic Chaos and (B) Stochastic Noise

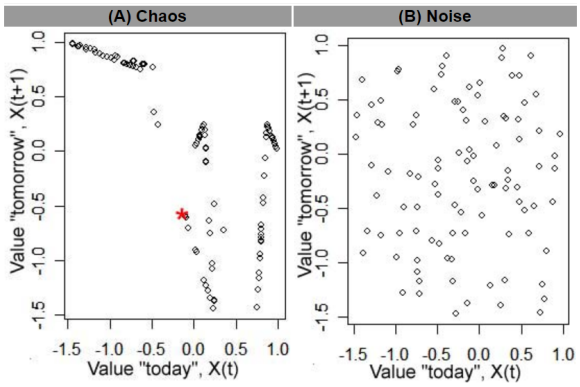


Figure 39. 2-Dimensional Phase Portrait Using Time Series Representing (A) Deterministic Chaos and (B) Stochastic Noise [17]

EDM techniques are also unable to correctly model systems which represent true *stochasticity* (also known as *high dimensional noise* or *true randomness*). Fig. 38 demonstrates this limitation using example *time series* representing (A) *deterministic chaos* and (B) *stochastic noise* [17]. It may be difficult to notice a significant difference between these *time series*. However, the 2-dimensional *phase portraits* of these *time series* shown in Fig. 39 show that there is clear auto-regressive behavior present in *time series* (A) as demonstrated by the distinct linear trends throughout the phase portrait. Identifying and quantifying such auto-regressive trends is the pur-

pose of Takens' Theorem which EDM techniques are based upon [13, 81, 83, 84, 120]. *Stochastic noise* lacking such trends makes it highly unlikely to find the correct dimensionality to use during *manifold* reconstruction, among other challenges. The term *manifold* refers specifically to 'a *compact manifold* M of dimension m' defined by 'a smooth one-parameter family of vector fields Z_Ω and a smooth one-parameter family of functions y_Ω' [81]. Generic examples of *compact manifolds* include a sphere, torus, double torus, cross surface, and Klein bottle [121].

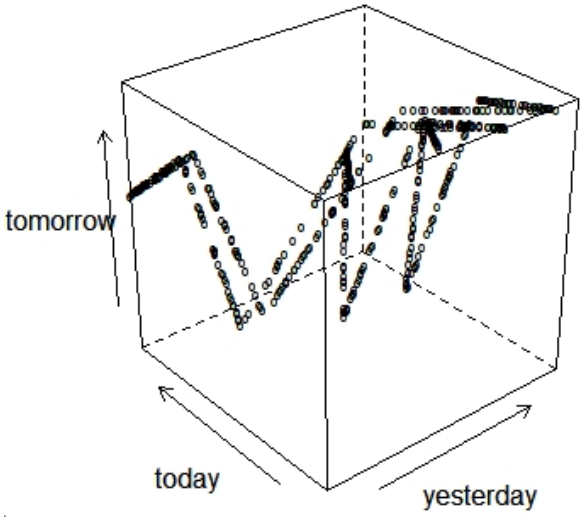


Figure 40. 3-Dimensional Phase Portrait Showing a Correctly Reconstructed Attractor Using Time Series Representing Deterministic Chaos [17]

Fig. 40 demonstrates the importance of selecting the correct dimensionality for *manifold* reconstruction. Fig. 40 is a 3-dimensional *phase portrait* of *time series* (A) from Fig. 38. Notice how every point in the *manifold* is now differentiable compared to the 2D reconstruction shown in Fig. 39. Some points may appear to overlap in the single perspective provided by Fig. 40; however, hopefully it is clear to the reader that viewing the 3-dimensional space from different angles allows for each point to be uniquely identified. Being able to distinguish between each vector $\in Z_\Omega$ is critical to accurately measure whether two *shadow manifolds* may be diffeomorphic (maps 1:1) during Convergent Cross Mapping (CCM). A group of non-differentiable portions of

manifold M due to an incorrect selection of dimensionality m to express M is referred to as a 'singularity'. Singularities obstruct the diffeomorphic relationship between *shadow manifolds* and may lead to misleading or incorrect findings by EDM methods. The concepts of *shadow manifolds*, diffeomorphism, and CCM were introduced in Section 2.7.

6.2 Selecting Embedding Dimension

Simplex Projection is an EDM method which may be used to make predictive models using manifold reconstruction. When the ultimate goal is to use Convergent Cross Mapping (CCM) and multi-view embedding to detect causal relationships and build predictive models, Simplex Projection may be used to estimate the optimal embedding dimension to use during manifold reconstruction. This is done by iteratively testing predictive accuracy at various embeddings. Once all 'singularities' in the manifold have been converted into differentiable manifold segments (as demonstrated between Fig. 39 and 40 in Section 6.1), prediction accuracy should be maximized. Thus, the optimal embedding factor E for *shadow manifold* reconstruction using a particular *time series* is the local maximum of the curve defined by embedding dimension and prediction accuracy ρ produced by iterative runs of Simplex Projection.

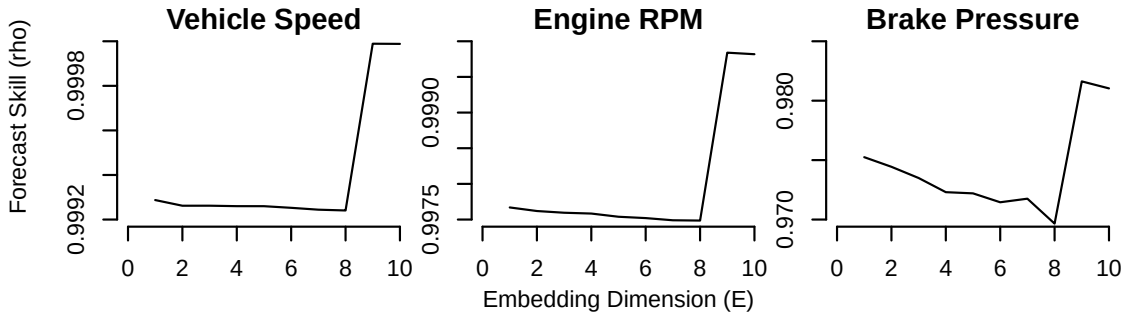


Figure 41. Simplex Projection Forecast Skill ρ With Respect to Embedding Dimension E (Controlled Driving Scenario)

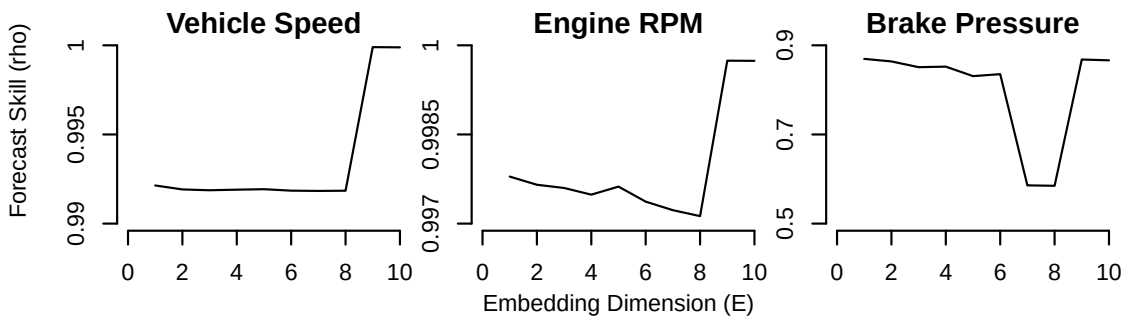


Figure 42. Simplex Projection Forecast Skill ρ With Respect to Embedding Dimension E ('City' Driving Scenario)

Fig. 41 and 42 show the results of performing Simplex Projection using multiple embedding factors with the *time series* for vehicle speed, engine RPM, and brake pressure. These figures represent the controlled and 'city' driving scenarios. Analysis of both driving scenarios and all three features resulted in nine as the optimal embedding dimensionality for *shadow manifold* reconstruction. This "optimal value does not have to correspond to the dimensionality of the original system... it is more useful to think of the embedding dimension as a practical measure that is dependent on the properties of the data" [85]. Furthermore, Whitey's Theorem asserts that the true number of variables affecting the *time series* in the actual system lies somewhere between E and $2 \times E + 1$ [17, 87]. For the remainder of this chapter, all techniques use a manifold embedding dimensionality of 9. This means that the *shadow manifold* for each feature is constructed using the original *time series* and eight copies of the same *time series* lagged by a multiple of a constant τ. Specifically, these extra dimensions are the set of *time series* lagged by $\{\tau, 2\tau, 3\tau, 4\tau, 5\tau, 6\tau, 7\tau, 8\tau\}$. A single payload offset in the chronologically ordered set of payloads X_{ID} is used as τ in the analysis presented in this chapter.

6.3 Characterizing System Linearity

Simplex Projection and Sequential Locally Weighted Global Linear Maps (S-map) may both be used to determine whether a system is globally linear. This analysis is not necessary to correctly reconstruct manifolds during CCM and multi-view embedding. However, like the *phase portraits* discussed earlier in this chapter, understanding whether a system is globally linear aids in the interpretation of results from CCM and multi-view embedding. For example, the plots of forecast skill ρ and embedding dimension shown in Section 6.2 had very minor differences in ρ as embedding dimension was adjusted. This scale was enlarged in the 'city' driving scenario compared to the controlled driving scenario. The scale was also larger for brake pressure compared to vehicle speed and engine RPM. This tight range of ρ is the result of the globally linear behavior exhibited by vehicle speed and engine RPM relative to brake pressure. It also highlights how a more diverse driving sample enable each feature to demonstrate more dynamic behavior. CCM analysis of manifolds constructed from globally linear features may be expected to result in an approximately linear strength of causal relationship as sample (library) size is increased.

Simplex Projection was iteratively run using $E = 9$ and increasing values of *time to prediction tp* to quantify the linearity of the vehicle speed, engine RPM, and brake pressure *time series* in the controlled and 'city' driving scenarios. *tp* is the difference in time between a point used to make a prediction about a future value and the future value. The results of this iterative Simplex Projection analysis for both driving scenarios are presented in Fig. 43 and 44. The x-axis for these figures is the number of payloads in the future used to test forecast skill ρ. More specifically, the *time* in *tp* is the number of payloads in the future used to make a forecast; this represents some length of time because payloads are chronologically ordered. 1,000 payloads equates to approximately 12 seconds of driving time, 600 payloads to 7 seconds, and

200 payloads to 2.5 seconds. The gradual drop in forecast skill ρ for vehicle speed and engine RPM indicate that they may be globally linear systems. While this does not preclude them from being used as part of EDM techniques, it does suggest that simpler computational models, such as Granger Causality or auto-regressive models, may be viable alternatives when developing models or performing *semantic analysis* [17, 84, 88, 89].

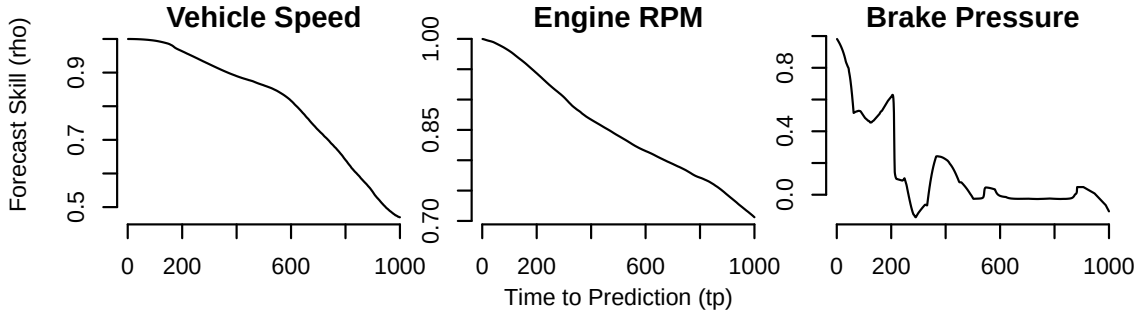


Figure 43. Simplex Projection Forecast Skill ρ With Respect to Time to Predict tp (Controlled Driving Scenario)

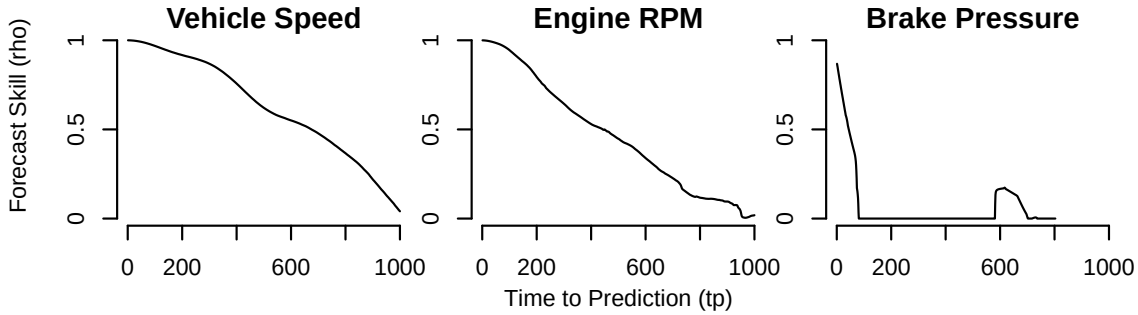


Figure 44. Simplex Projection Forecast Skill ρ With Respect to Time to Predict tp ('City' Driving Scenario)

Sequential locally weighted global linear maps (S-map) may be used in a similar fashion to quantify system linearity. S-map differs from Simplex Projection in that a non-linear tuning parameter θ is adjusted instead of tp. S-map analysis of globally linear systems is maximized when θ is 0. This relationship may be intuitively explained using the *phase portraits* shown in Fig. 35, 36, 37, and 40. The points in

36 and 37 tightly fall along a single distinct vector. Thus, predicting the location of a single point along that vector may be accurately achieved using *all* other points in the manifold ($\theta = 0$). Conversely, the points in Fig. 35 also trend along a single vector but also exists off of that axis. This means forecast skill ρ may improve using a subset of points closest to the point being predicted ($\theta > 0$). This scenario is even more apparent in Fig. 36 which has very distinct local vectors defining behavior at various parts of the manifold.

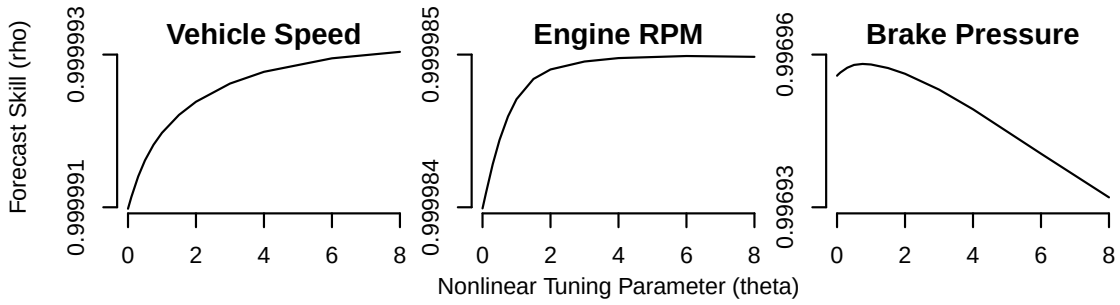


Figure 45. S-map Forecast Skill ρ With Respect to the Nonlinear Tuning Parameter θ (Controlled Driving Scenario)

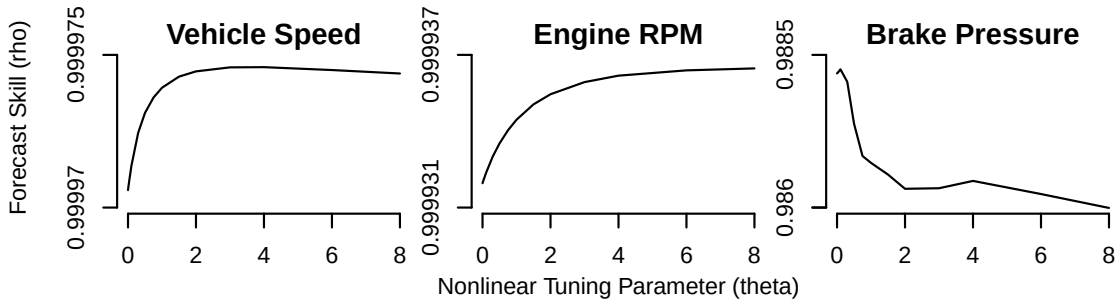


Figure 46. S-map Forecast Skill ρ With Respect to the Nonlinear Tuning Parameter θ ('City' Driving Scenario)

Fig. 45 and 46 present the result of iterative S-map forecast skill ρ at various levels for the non-linear tuning parameter θ. As with Simplex Projection, the small range of differences in ρ for the controlled driving scenario and vehicle speed and engine RPM features is evidence of globally linear behavior.

6.4 Quantifying Causality

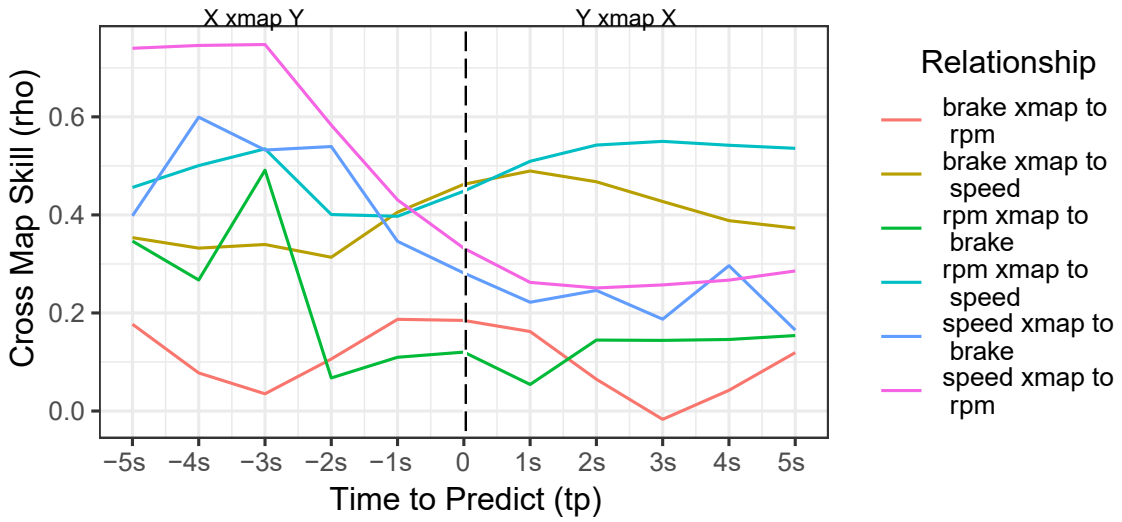


Figure 47. Convergent Cross Mapping Forecast Skill ρ With Respect to Time to Predict tp ('City' Driving Scenario)

The strength of causal relationships between vehicle speed, engine RPM, and brake pressure is summarized in Fig. 47. The legend lists a series of X *xmap* Y relationships. The term X *xmap* Y refers to the CCM analysis which used the shadow manifold M_X to make forecasts onto the shadow manifold M_Y. Somewhat counter-intuitively, the resulting line of cross map skill ρ given time to predict tp is then interpreted as the strength of the causal effect of Y onto X. Higher predictive skill ρ indicates a stronger causal effect of Y onto X when time to predict tp is positive. Time to predict tp is the delay—or *lag*—from the time a change in feature Y has a causal effect manifest in the feature X. Since cause must precede effect, negative values of tp effectively reverse the interpretation of X *xmap* Y from Y *causes* X to X *causes* Y. Fig. 47 allows the following observations to be made:

1. Brake pressure has a weak and nearly instant causal effect on engine RPM ($\rho \approxeq 0.19$)

2. Speed has a moderate causal effect on engine RPM after an approximately 1 second delay ($\rho \cong 0.49$)

3. Engine RPM has a moderate causal effect on brake pressure after an approximately 3 second delay ($\rho \cong 0.49$)

4. Engine RPM has a moderate causal effect on vehicle speed after an approximately 3 second delay ($\rho \cong 0.53$)

    - Note: *rpm xmap speed* also suggests the reverse relationship is true as well

5. Speed has a strong causal effect on brake pressure after an approximately 4 second delay ($\rho \cong 0.60$)

6. Speed has a strong causal effect on engine RPM after an approximately 3 to 5 second delay ($\rho \cong 0.75$)

Fig. 48 uses a graph format to expresses the causal relationships between the vehicle speed, engine RPM, and brake pressure features found in Fig. 47. The arrows pointing from feature X into feature Y represent a causal relationship from X onto Y. Each arrow has two Pearson's ρ values listed next to it using blue and red text. The two colors represent the two manifolds—X xmap Y and Y xmap X—used to quantify causal relationships between X and Y during CCM analysis. Each of these cross mapping results were found be looking for the peaks of each line in Fig. 47. The times listed under each ρ is the time to predict tp on the x-axis associated with a particular line's peak. A range of tp values is listed whenever the curve segment adjacent to the maximum cross mapping skill ρ remained subjectively close to the maximum. This *plateau* of the strongest observed cross map skill ρ defined by the

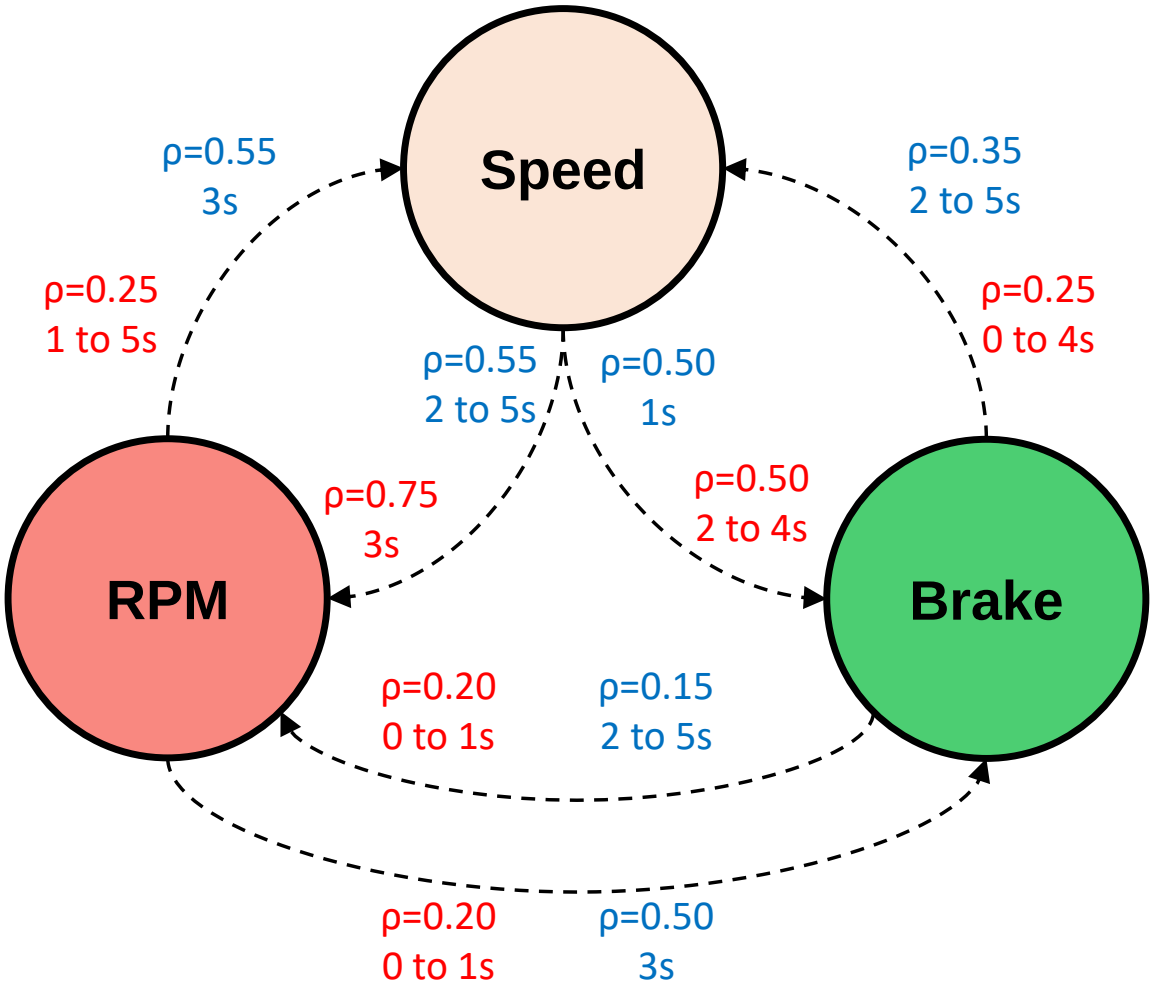


Figure 48. Convergent Cross Mapping Forecast Skill ρ and Time to Predict tp Expressed as a Graph ('City' Driving Scenario)

range of tp values on the x-axis is intended to represent the range of times the causal physical process was observed to cause an effect on affected process.

Ideally, approximately the same maximum cross map skill ρ and its corresponding time to predict tp are observed regardless of which *shadow manifold* is used during CCM. If the results significantly differ based on the *shadow manifold* used then there is uncertainty about how to interpret the dissimilar findings. Such uncertainty could mean that CCM may not be viable for quantifying causal relationships between automotive CPS. The comparisons shown in Fig. 48 suggest that this is not the case. Comparing the red and blue results for each causal relationship reveals a consistent

maximum ρ and corresponding tp regardless of the *shadow manifold* used during CCM. Each pair of findings presented using red and blue text next to each directed causal relationship reveals overlapping time to predict tp ranges for the maximal observed cross map skill ρ. The only counter example of this trend is the causal relationship from RPM onto brake pressure. Findings from the RPM *shadow manifold* had a different tp and ρ result compared to the results from using the Brake Pressure *shadow manifold*.

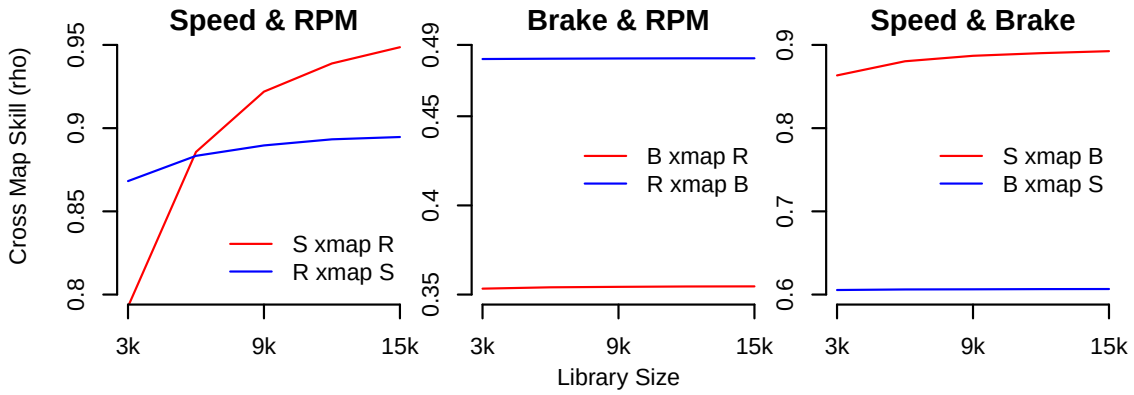


Figure 49. Convergent Cross Mapping Forecast Skill ρ With Respect to Library Size (Controlled Driving Scenario)

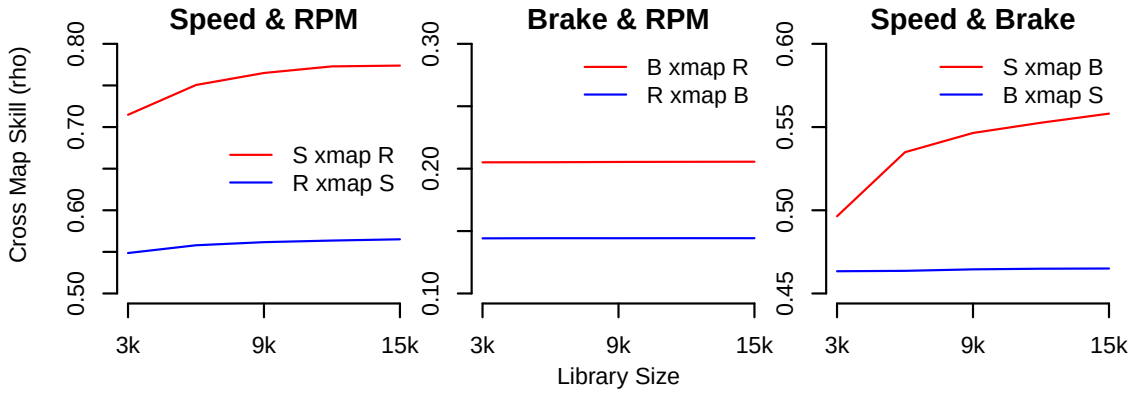


Figure 50. Convergent Cross Mapping Forecast Skill ρ With Respect to Library Size ('City' Driving Scenario)

Section 6.1 discussed the limitation that CCM may reflect causal relationships

which are idiosyncratic to the particular set of driving conditions. Using the controlled driving scenario described in Section 3.2 as an example, braking pressure was only applied at low speed and low engine RPM. CCM analysis of that driving scenario is presented in Fig. 49. The maximal cross map skill ρ presented in the middle plot of Fig. 49 suggests that engine RPM has a relatively weak causal relationship onto brake pressure compared to brake causing RPM. Fig. 50, based on the 'city' driving scenario, flips these relative strengths and significantly lowers both. This inconsistent pair of findings mirrors the conflicted results highlighted at the conclusion of the previous paragraph discussing Fig. 48.

It is unclear why only the relationships quantified by using the engine RPM *shadow manifold* to cross map onto brake pressure were the only ones to suffer from inconsistent results when comparing different *shadow manifolds* and driving scenarios. This inconsistency may be caused by the fact that the driving samples were collected using vehicle 14 which is a high efficiency hybrid sedan. The combination of the hybrid system dynamically turning off the internal combustion engine, the driver's dynamic decision making on when to apply braking and acceleration, and the relatively brief sample duration of approximately four minutes may all be contributing factors that led to an inconsistent observed relationship between engine RPM and braking pressure. Inconsistent findings may be avoidable if experimental design best practices are explored using targeted research designed to test when inconclusive CCM results do and do not occur given different combinations of vehicles, drivers, driving scenarios, and sample duration.

Subtle Causal Relationships.

Some of these CCM results may initially seem to disprove the method's viability in the context of automotive network analysis. For example, considering the vehicle

network from a purely mechanical point of view may lead some to find the result that engine RPM or vehicle speed has a relatively strong causal effect on brake pressure preposterous. However, deeper consideration of these results in particular actually highlights the power of EDM in the context of *semantic analysis* of cyber-physical systems (CPS).

There is certainly no *mechanical* causal relationship from current vehicle speed or engine RPM onto future brake pressure; only the driver (and to a small degree some special systems like Anti-Lock Brakes) control brake pressure. Therein lies the deeper finding being made by this CCM analysis. The *driver* knows they need to come to a stop or significantly slow down in the near future. This causes the driver to decrease pressure on the accelerator (and consequently reduce acceleration and engine RPM) in anticipation of the braking event. Soon thereafter they transition to applying brake pressure. For this particular driver, CCM analysis indicates that these transition events took approximately 3 seconds. Thus *there is* a clear causal relationship from a change in vehicle speed and engine RPM onto future brake pressure conveyed not through mechanical means, but through the driver.

6.5 Comparing Prediction Accuracy to Statistical Methods

This section briefly highlights the predictive power of one EDM technique—Multi-view embedding (MVE)—compared to a statistical prediction method like ordinary least squares linear regression. Section 6.3 highlighted the result that vehicle speed, engine RPM, and brake pressure appear to exhibit highly linear system behavior. The phase plots of vehicle speed, engine RPM, and braking pressure shown in Fig. 35, 36, and 37 visually demonstrate the linear system behavior of these features. These findings motivated the selection of linear regression as the base case for a statistical modeling technique to compare with an EDM alternative. Brake pressure was selected

as the response variable being predicted by the other two features because of its relatively non-linear behavior identified by simplex projection, S-map, and visual analysis of 3D phase plots.

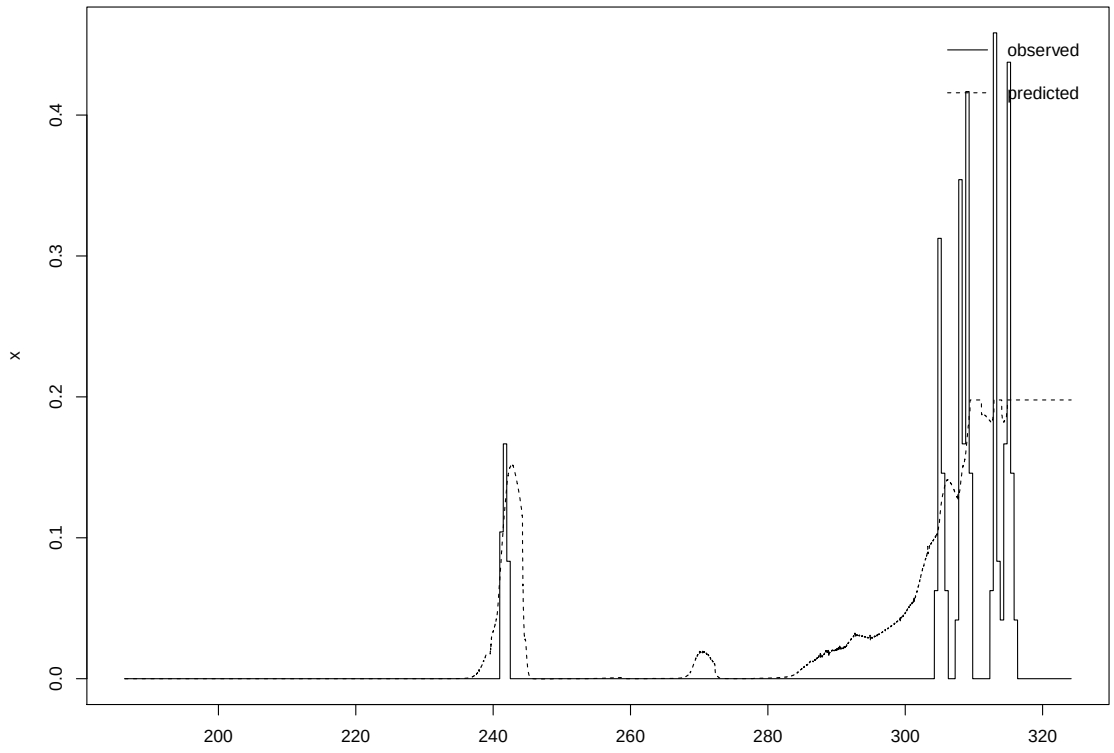


Figure 51. Actual Brake Pressure and Predicted Brake Pressure Using Ordinary Least Squares Linear Regression ('City' Driving Scenario)

Fig. 51 presents the actual brake pressure over time over time in a 'city' driving sample as a solid black curve. The dashed black curve is the predicted brake pressure over time using linear regression with vehicle speed and engine RPM as the independent variables. The coefficient of determination R^2 for this model was 0.102. This may be interpreted that the linear regression model could only explain approximately 10% of the variance in braking pressure.

Fig. 52 presents the result using the same sample with MVE [53, 85, 89]. MVE

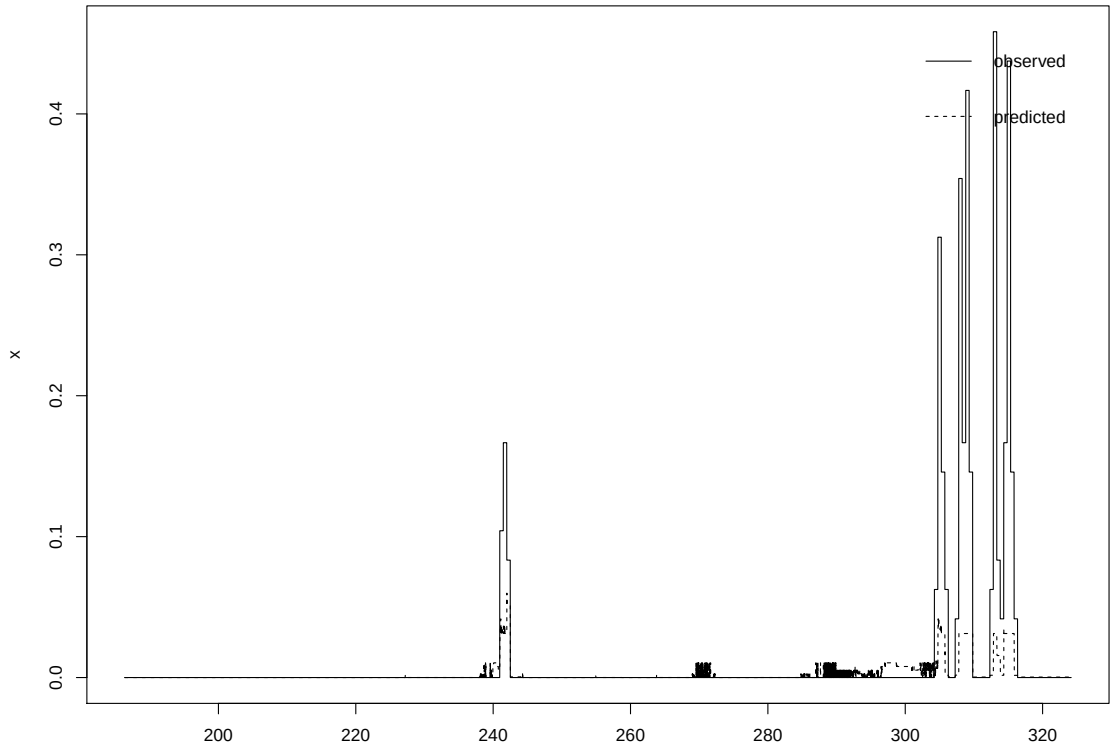


Figure 52. Actual Brake Pressure and Predicted Brake Pressure Using Multiveiw Embedding ('City' Driving Scenario)

works similarly to CCM except more than one feature and their lags are used during manifold reconstruction. In this case only the vehicle speed and engine RPM *time series* were used to create the manifold. The dashed black curve is the predicted brake pressure over time using the MVE manifold. The coefficient of determination R^2 for this model was 0.61. This represents just over a 50% increase in variance explained compared to linear regression.

6.6 Conclusions

The ability to quantify the subtle, indirect, but nevertheless important causal relationships coupled with the largely unsupervised nature of EDM methods make

them particularly well suited for CPS *semantic analysis* and IDS research. Because EDM is uniquely capable of detecting causal relationships present in dynamic systems, it provides a natural compliment to the *semantic analysis* techniques presented in Chapter V. These two *semantic analysis* techniques provide a wealth of information about the relationships among *time series*. We believe that additional *semantic analysis* is likely possible by studying a graph structure created using *signals* clusters as nodes and causal relationships between clusters as edges. Such graph based analysis may greatly extend the utility of *label propagation* by enabling inductive labeling of unlabeled *signal* clusters based on their causal relationships to labeled *clusters*.

VII. Conclusions

This dissertation motivated the need to automate *lexical* and *semantic analysis* for Cyber-Physical System (CPS) networks such as the Controller Area Network (CAN) bus used in passenger vehicles. Chapter IV presented techniques for automated *lexical analysis* intended to extract *time series* from proprietary payload formats. Automated *semantic analysis* of those payloads included finding correlated relationships in Chapter V and causal relationship in Chapter VI. An array of validation metrics and strategies were presented for each type of analysis.

All three focus areas—*lexical analysis*, *semantic analysis*, and validation—represent significant novel contributions to the field of automated analysis of proprietary CPS networks. These unsupervised techniques are intended to be a springboard for the independent cyber-security research community. They are expected to be of direct and immediate use to the cyber-security auditing community. This is because they represent a *very* educated initial guess for the correct payload *tokenization* and functional mapping of between *time series* present in the vehicle network. Auditors can quickly and easily tweak this guess as needed until they are confident that the payloads have been correctly reverse engineered. This reduces the time required for manual reverse engineering from days down to hours or even minutes.

The research community is likewise expected to benefit by having a robust set of techniques and validation strategies to improve future research. A consistent challenge for research in the CPS network domain is access to truth data for validating techniques and findings. The comparison testing strategies presented in this paper may be of direct use in future research or at least inspire similar research on how to perform validation without truth data. The successful application of techniques from Life Sciences research may also serve as inspiration to look beyond statistical machine learning research.

The output of these techniques may also enable the efficient generation of large and diverse labeled data sets. Access to such data sets is expected to enable the development of superior automated *lexical* and *semantic* analysis of proprietary networks using supervised machine learning methods. Labeled data sets may also significantly aid research in robust CPS network simulation techniques. Until that time, these techniques may serve as a reliable 'gold standard' for computer aided analysis of proprietary non-text network protocols.

VIII. Future Work

The relatively new and compelling field of Empirical Data Modeling (EDM) presents a significantly opportunity to improve the state of the art in semi-supervised machine learning. For example, the semi-supervised learning pipeline proposed by Glennan et al. was discussed in Section 2.5 and presented in Fig. 12. That pipeline represented a sequence of unsupervised clustering to produce a fully labeled data set, supervised classification, and then another unsupervised clustering phase to implement a fuzzy classification strategy. It is expected that some data may remain effectively unclassified (using the 'unknown' label) if very few or no labeled data are present in a particular cluster during either clustering step. Using EDM in the context of CPS network *semantic analysis* where the majority of data are expected to be *time series*, it may be possible to leverage causal relationships between clusters to classify more of the remaining unlabeled data. To demonstrate this, consider Fig. 53 which presents the output of a hypothetical semi-supervised data set labeling process.

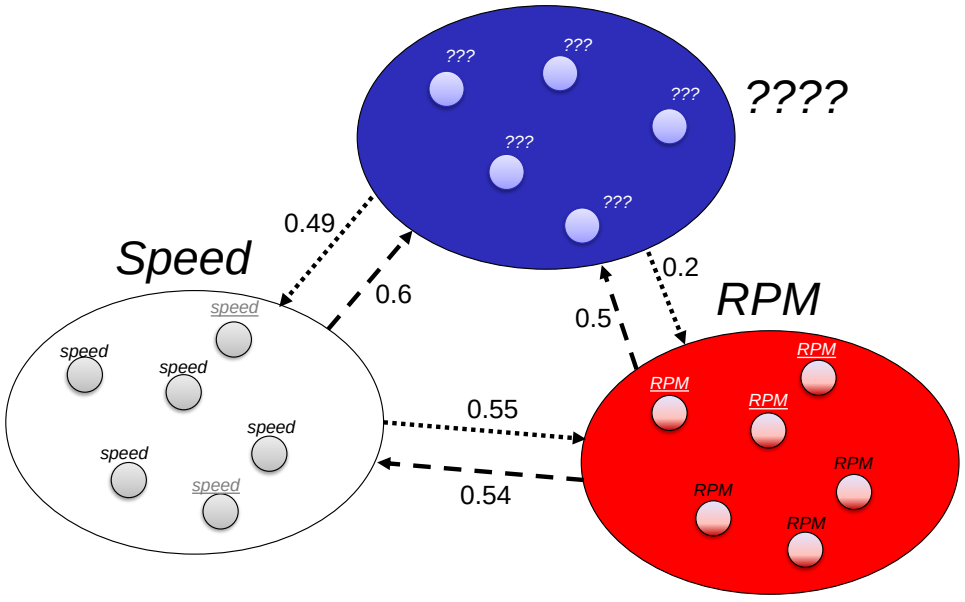


Figure 53. Example Result of Semi-Supervised Data Set Labeling

The points in the Speed and RPM clusters with black text represent the originally

labeled *time series*. In the case of automotive CAN network analysis, these labels may have come from correlation with limited J9179 diagnostic information. The other points in these two clusters with underlined grey or white text represent additional *time series* that were labeled by being in the same cluster of correlated *time series*. This process is referred to as *label propagation* in semi-supervised machine learning research [9, 10, 19, 80].

Current semi-supervised machine learning proposals have no reasonable means to label the points in the blue cluster. This is because they don't have the benefit of knowing causal relationships between clusters which is possible using EDM with *time series* data sets. The graph structure produced by adding causal links between clusters along with some limited manual reverse engineering may sufficient to train a supervised classifier to identify the blue cluster. For example, manual reverse engineering several vehicles may reveal that brake pressure *signals* consistently produce a cluster with the relative strengths of causal relationships to vehicle speed and engine RPM clusters as shown in Fig. 53[1]. These labeled weighted directed graphs could be used to train a supervised classifier to identify one or more of these clusters based on the graph structure and labeled clusters. A hypothetical result of this graph based cluster classification is shown in Fig. 54.

The weighted directed graph and labeled clusters might be iteratively passed as input to the classifier to label more clusters much like how the game of Sudoku is played. For example, imagine that there are several more unlabeled clusters than those shown in Fig. 54 with corresponding observed causal relationships between each pair of clusters. The classifier may not be confident in its classification of those other clusters using only the Speed and RPM cluster labels; however, it *is* confident in its

[1]The decision on exactly how the measure the strength of causation between *time series* in one cluster to those in another is very similar to selecting a *linkage* strategy during Agglomerative Hierarchical clustering

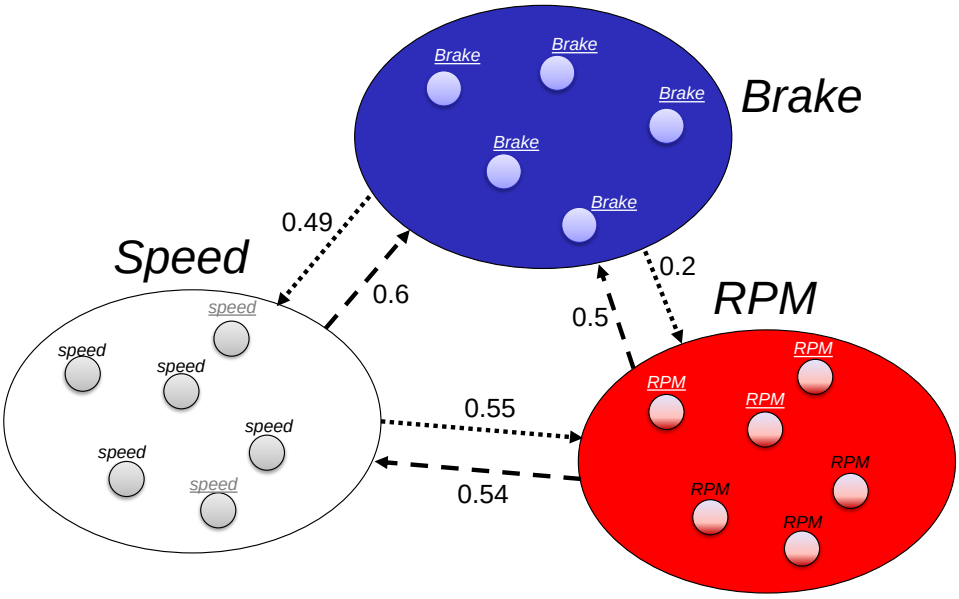


Figure 54. Example Result of Semi-Supervised Data Set Labeling Augmented With Graph Based Classification

classification of the brake cluster. The same graph is passed back to the classifier but now three clusters—Speed, RPM, and Brake—are labeled. Assume this third label increases the confidence for labeling one or more of the other unlabeled clusters of *time series* above a minimum threshold. This process is repeated until no new labels are assigned. At which point the researcher might decide to manually reverse engineer the remaining clusters and re-train the classifier. This process of iterative improving the graph based classifier might continue until the entire vehicle CAN bus can be accurately classified using the a few J1979 diagnostics and the weighted directed graph produced using the techniques described in this paper. Such a result would hopefully encourage manufacturers to abandon their policy of *security through obscurity* in favor of legitimate cyber-security mechanisms.

Appendix A. Additional Background Information on Automated Network Traffic Reverse Engineering

This dissertation is not the only work to recognize the applicability of life science techniques to the field of automated network traffic reverse engineering. In his 2004 *Toorcon* presentation, Beddoe highlighted the similarities between bioinformatics and protocol reverse engineering [122]. His presentation and self published paper proposed using the Needleman Wunsch algorithm, the Smith Waterman algorithm[123], the Unweighted Pairwise Mean by Arithmetic Averages (UPGMA) algorithm, and Phylogenetic Trees developed within the bioinformatics community to reverse engineer Microsoft's SMB protocol [124] . In addition to the presentation and paper, Beddoe also uploaded his prototype code for using these ideas to reverse engineer SMB to a public GitHub repository[1]. Beddoe does not appear to have published any evaluation of his proposals.

Weng et al. also explored the idea of using the Needlman-Wunsch algorithm as a component in their 2012 conference paper "A semantics aware approach to automated reverse engineering unknown protocols" [52]. Their approach is to iteratively consider various n-grams of various byte lengths given a string generated by a stateful application layer protocol. They then used the bottleneck clustering algorithm to group similar n-grams based upon bespoke heuristics about the contents of the n-grams. Finally, they employed the Needleman-Wunsch algorithm to align and combine the n-grams within a cluster to produce a final candidate for a lexical *token*. Several other papers also considered the use of unsupervised clustering as part of an automated reverse engineering approach for a stateful, text based, variable packet length, application layer protocols such as Server Message Block (SMB) and File Transfer Protocol (FTP) [50, 51, 125, 126, 127].

[1] https://github.com/unmarshal/protocol-informatics

The use of taint analysis also appears to be a popular method for achieving automated reverse engineering of network protocols [95, 128, 129, 130, 131]. Taint analysis is essentially accessing to the memory being used by a process communicating with the target protocol to infer a finite state machine that conforms to the protocol standard as well as the message format used in the protocol. We propose that direct access to the memory in each of a vehicle's ECUs is not a realistic data collection method so taint analysis are not considered further. However, taint analysis may be effective in the context of the medical electronics industry which may have relatively fewer models and ECUs per machine than the automotive industry.

DeYoung, Puupera, Roning, and Antunes et al. present methods for inferring a finite automata compliant with an observed application layer network trace using methods from the field of computational linguistics [50, 51, 125, 132]. Puupera used the concept of local search using the idea of *suffix arrays*. Unfortunately, the authors realized that their approach was too simplistic for their goal of inferring automata from variable length network protocols and no meaningful results were pursued. Similar work by the same research team based on context free grammars continue as part of the PROTOS project [51].

Three proposals by Choi et al., Trifilo et al., and Cui et al. featured methods for reverse engineering protocol format using a combination of statistical analysis and machine learning [18, 49, 133]. Trifilio et al. assumed a *payload* should be *tokenized* into one byte *signals*. The authors then looked for the variance of each *signal* as an indication for which were *static* protocol keywords. They applied bespoke heuristics for determining which levels of variance defined *static* and *dynamic signals* when generating a protocol automata. Choi et al. presented a non-text protocol reverse engineering effort also relying on the assumption that sub-flows are one or more bytes of data [49]. The approach ultimately leveraged a series of custom heuristic filters

to achieve approximately 40% accuracy in predicting the sub-flows used in a custom IEEE 802.15.4 protocol. In a separate paper, Cui et al. relied upon predictable text delimiters like spaces and tabs for text portions of observed network traffic and one byte segments for non-text network traffic to perform *lexical analysis* [18]. *Tokens* were clustered based on 20 statistical properties similar to those presented in Table 5. A second round of clustering with additional features was done to identified sub-clusters within the larger clusters identified during the first iteration. Then, any pairs of *tokens* with potential dependencies (such as a numerical *token* followed by a text *token* of that length) are merged. Afterwards, *tokens* in the same sub-clusters with significantly similar statistical properties are 'merged' into a single *token*. This process continues until all dependent and similar *tokens* are merged. Finally, the Needleman-Wunsch algorithm is used to align the remaining *tokens* with overlaps using heuristic distance measurements. This pipeline is presented in Fig. 55.

The 1997 conference paper 'Segmentation of expository texts by hierarchical agglomerative hierarchical clustering' by Yaari et al. demonstrated the effectiveness of agglomerative clustering for accurate *tokenization* of natural language text [30]. While their effort leveraged a known grammar being used by the corpus of text being segmented, it nevertheless demonstrates the effectiveness of agglomerative hierarchical clustering in a segmentation effort of similar complexity to CAN payload *tokenization*.

Muter's 'Entropy-based anomaly detection for in-vehicle networks' demonstrates the effectiveness of using entropy as a statistical feature for *signal* comparison in the automotive network domain. Muter's approach was to calculate the entropy of *signals* at the bit level, arbitration ID level, and total network level. A notion of 'distance' between *signals* was then calculated using the probability distributions of their entropy, $p(x)$ and $q(x)$ respectively, to calculate the relative entropy as shown

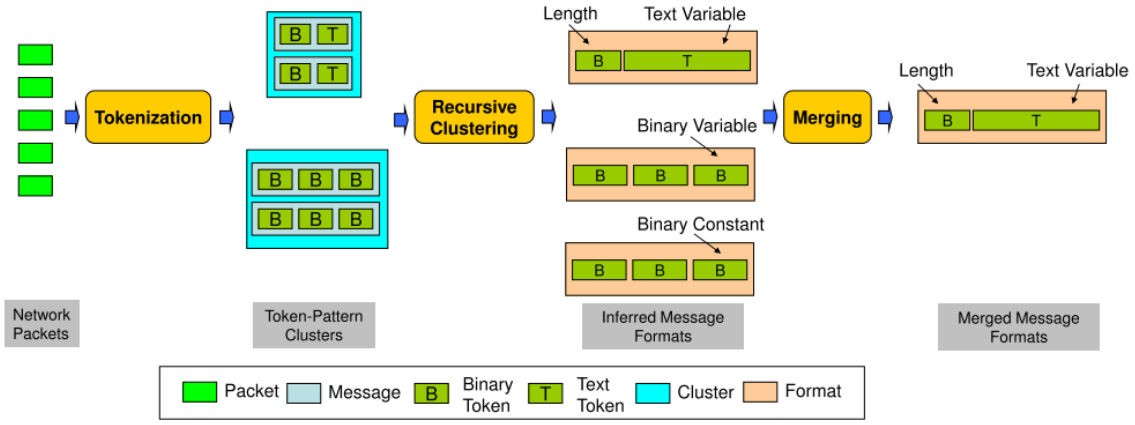


Figure 55. Overview of Discoverer's architecture presented as Fig. 1 in [18]

in equation 8.

$$RelEnt(p/q) = \sum_{x \in C_x} p(x) log \frac{p(x)}{q(x)} \tag{8}$$

Muter then used the levels of entropy and cross entropy to establish a notion of *vehicle state*. The insight proposed is that a parked vehicle generates *signals* which have much lower entropy than when the vehicle is moving. Thus, establishing ranges of entropy and cross entropy for each *signal* in a CAN network given different driving conditions enables the creation of a model describing the vehicle network that is simultaneously insensitive to 'normal' fluctuations in *signals* while being sensitive to deviations in entropy caused by new data caused by an attack. Deviations which exceeded a heuristic threshold were considered an attack. This method was able to detect a simulated attack on vehicle speed related *signals* for any value greater than ± 1 kph from the legitimate speed related *signals*.

In the context of classifying network *signals*, the term *semi-supervised classification* is defined by a majority of published proposals as training a supervised classifier that somehow makes use of unsupervised clustering before or after the classification process. This is done to expand the number of labeled classes and to improve clas-

sification accuracy. The main differentiation between the approaches is the choice of clustering algorithm, classification algorithm, and distance metrics.

Two clustering algorithms used by semi-supervised reverse engineering proposals are DBSCAN and k-means [10, 19, 80, 134, 135]. The important difference between these two methods is k-means guarantees every data point is added to a cluster. DBSCAN treats outlying points as *noise* which can remained unclustered in the final result. With the exception of the proposal by Ducange et al., these proposals all relied upon the k-Nearest Neighbors (k-NN) classifier. Ducange et al. instead chose to use a multi-objective fuzzy classifier [136]. k-NN may be the more common choice because of its consistent accuracy, simplicity, and computational complexity [80, 137].

The specific classifier used by Ducange et al. is a Multi-objective Evolutionary Fuzzy Classifier (MOEFC) which combined a Multi-Objective Evolutionary Algorithm (MOEA) and Fuzzy Rule-based Classifier (FRBC). The FBRC relies on a rule base(RB) which is a database of the decision rules used to classify data. A decision rule in this context is essentially a feature and threshold for separating classes. The MOEFC simultaneously optimizes the total rule length(TRL) of the RB and the FBRC's classification accuracy. In other words, an excessive TRL (the number of propositions used in the antecedents of the RB) is over-fitting the classifier and reduces the ease of interpreting the classification decisions. The balance between TRL and classifier accuracy is made using the Pareto Archived Evolution Strategy(PAES) rule and condition selection (RCS) algorithm [138]. Ducange et al. note that they used the "2+2" variant of the M-PEAS algorithm presented in [139].

These papers followed two trends for using statistical features and distance metrics. These trends are to either use one specific measurement like cross entropy and vector cosine similarity or a 'kitchen sink' approach of using as many features as possible [10, 9, 80, 19, 134, 135]. An example of the types and number of statistics used

Feature	Description	count
Duration	Duration of bidirectional flow	1
Packets	Number of packets transferred in each direction	2
Bytes	Volume of bytes transferred in each direction	2
Packet Size	Min., Max., Mean and Std Dev. of packet size in each direction	8
Inter Packet Time	Min., Max., Mean and Std Dev. of inter packet time in each direction	8
	Total	21

Figure 56. A set of statistical network traffic features used by [19]

in a 'kitchen sink' approach is shown in Fig. 56. Glennan et al. note that the 'kitchen sink' approach has the drawback that some features may actually inhibit classification performance [9]. To account for this, they incorporated feature selection as part of their semi-supervised proposal in order to select a subset of the 40 features they initially consider.

Appendix B. Agglomerative Hierarchical Clustering Dendrograms

The following Dendrograms provide a visualization of the clustering distance between *signal* clusters for each vehicle studied. The horizontal portion of each upside down U shape corresponds to the distance between clusters specified by level listed on the y-axis. The x-axis is unlabeled but corresponds to the individual *signals* observed in the vehicle's CAN bus network. The unique *signal* identifiers have been omitted from these figures to prevent clutter. A gray dashed horizontal line indicates the 0.2 maximum inter-cluster threshold used during single linkage Agglomerative Hierarchical Clustering.

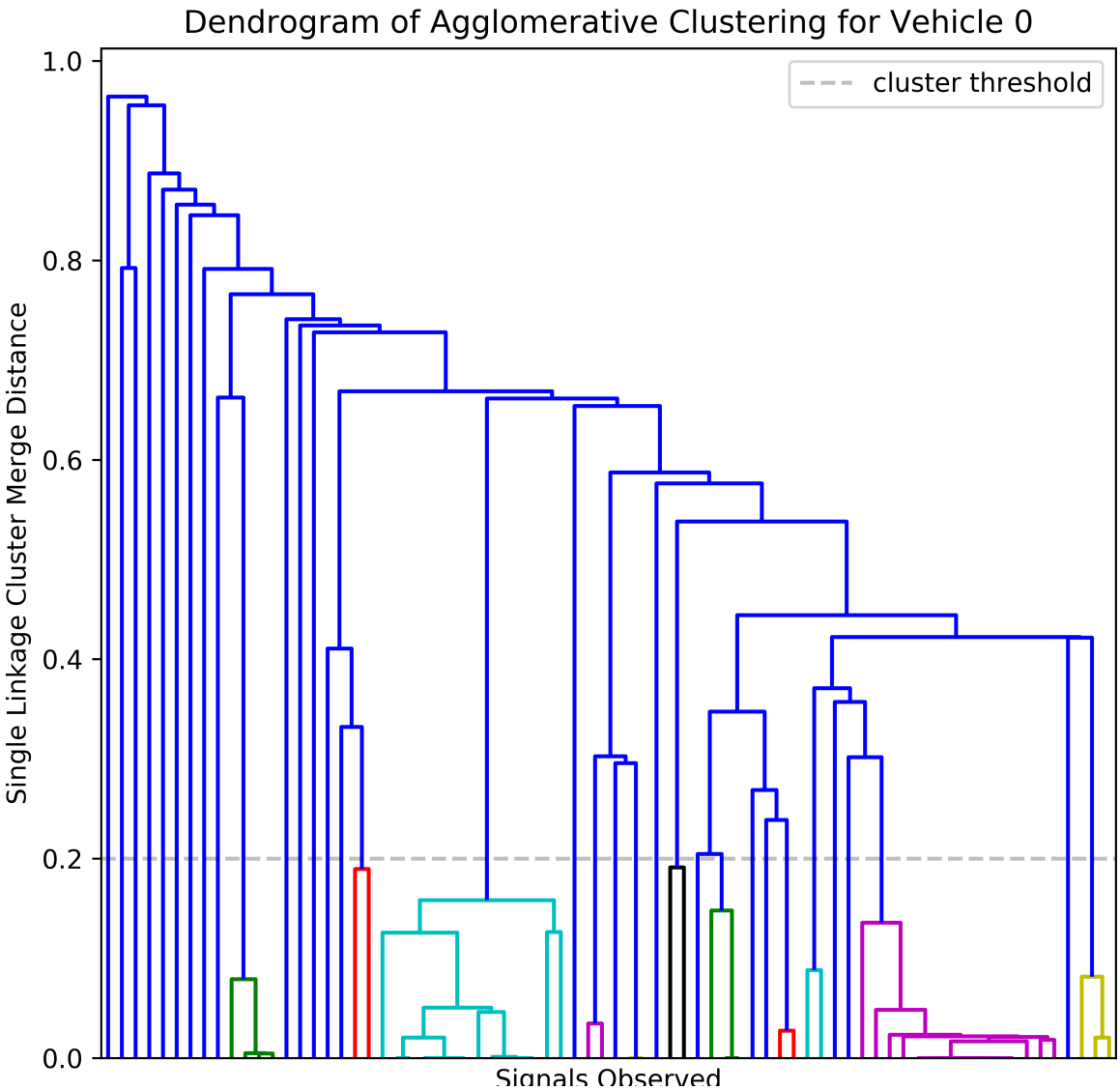


Figure 57. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 0

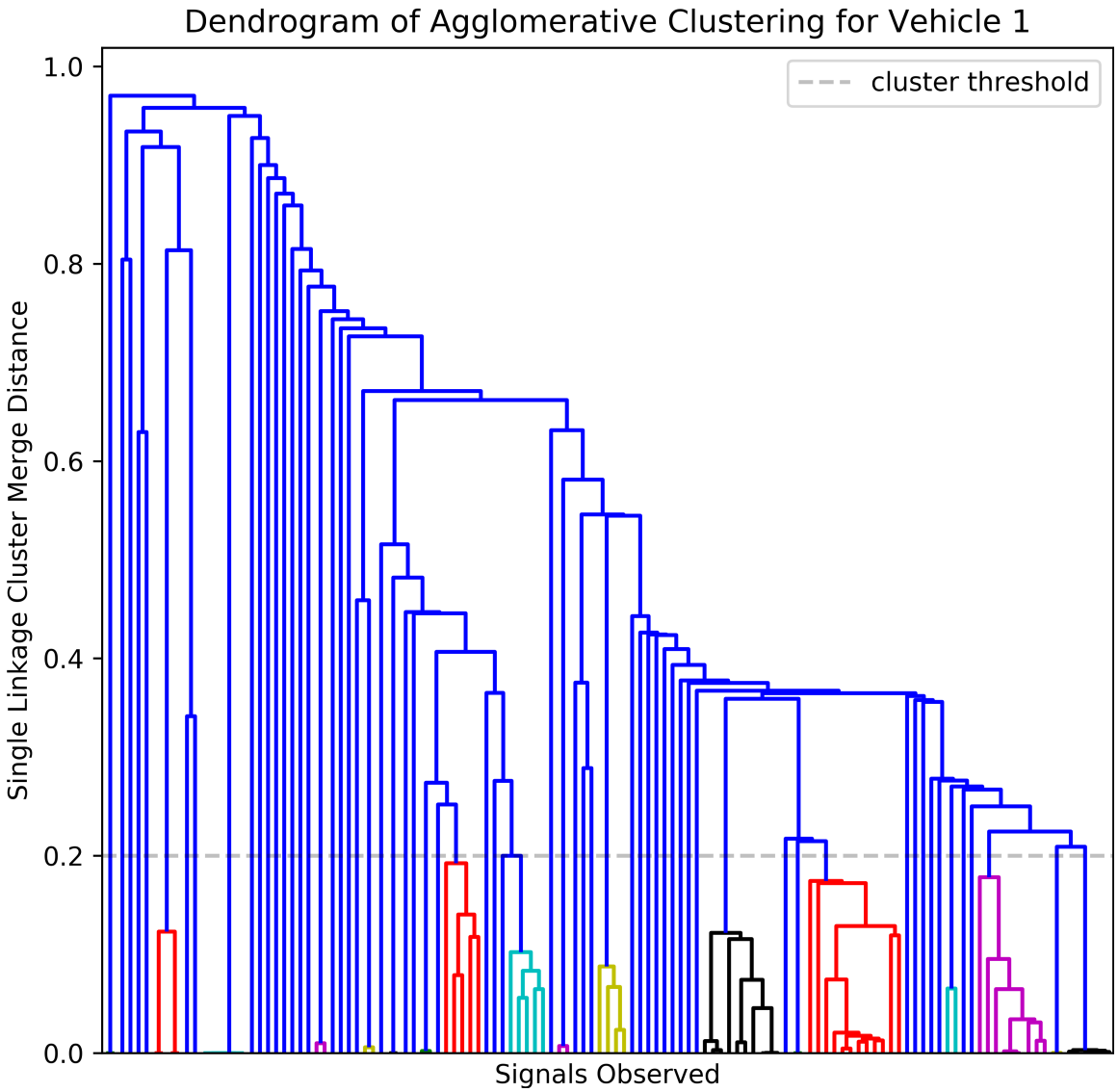


Figure 58. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 1

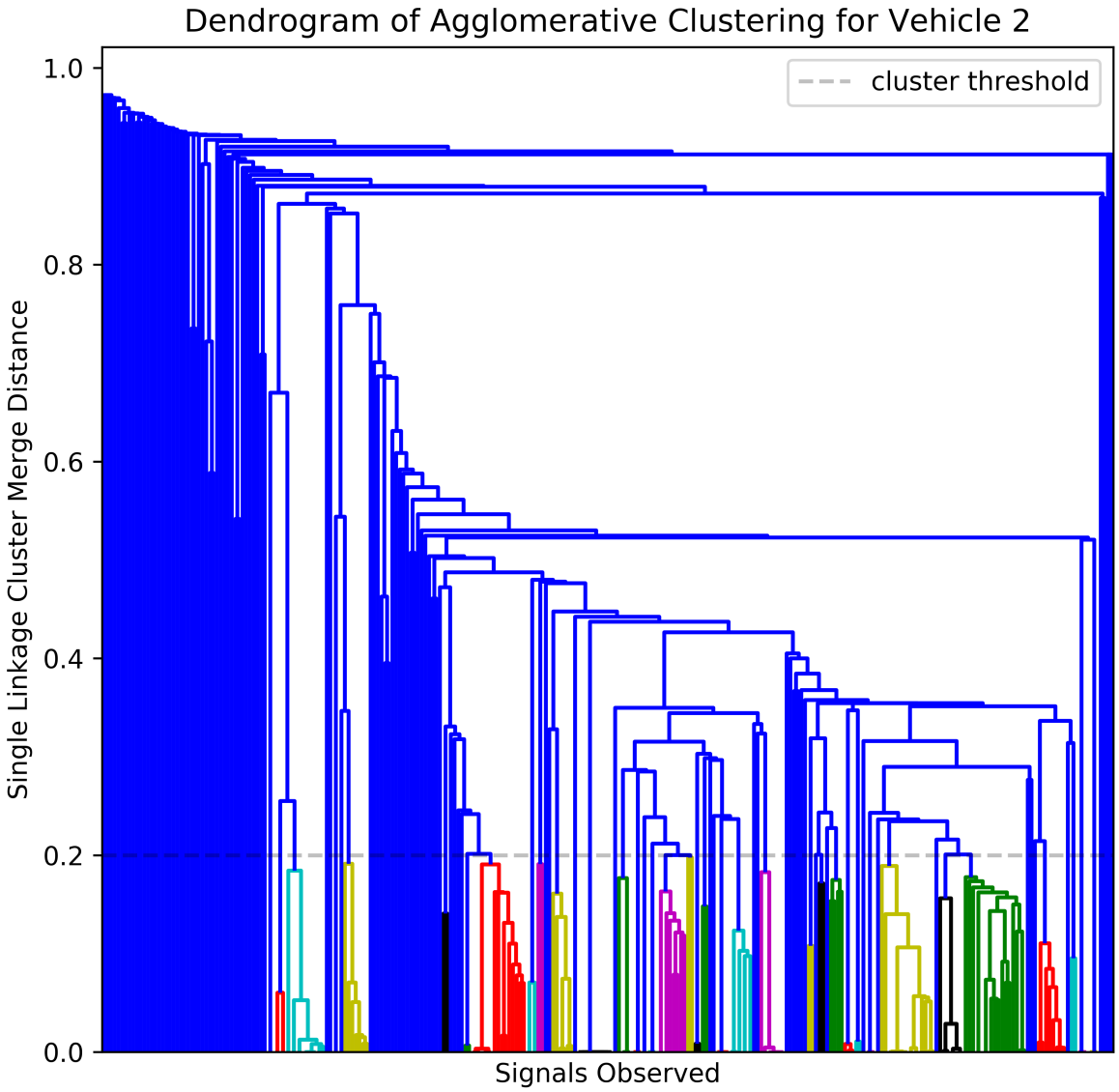


Figure 59. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 2

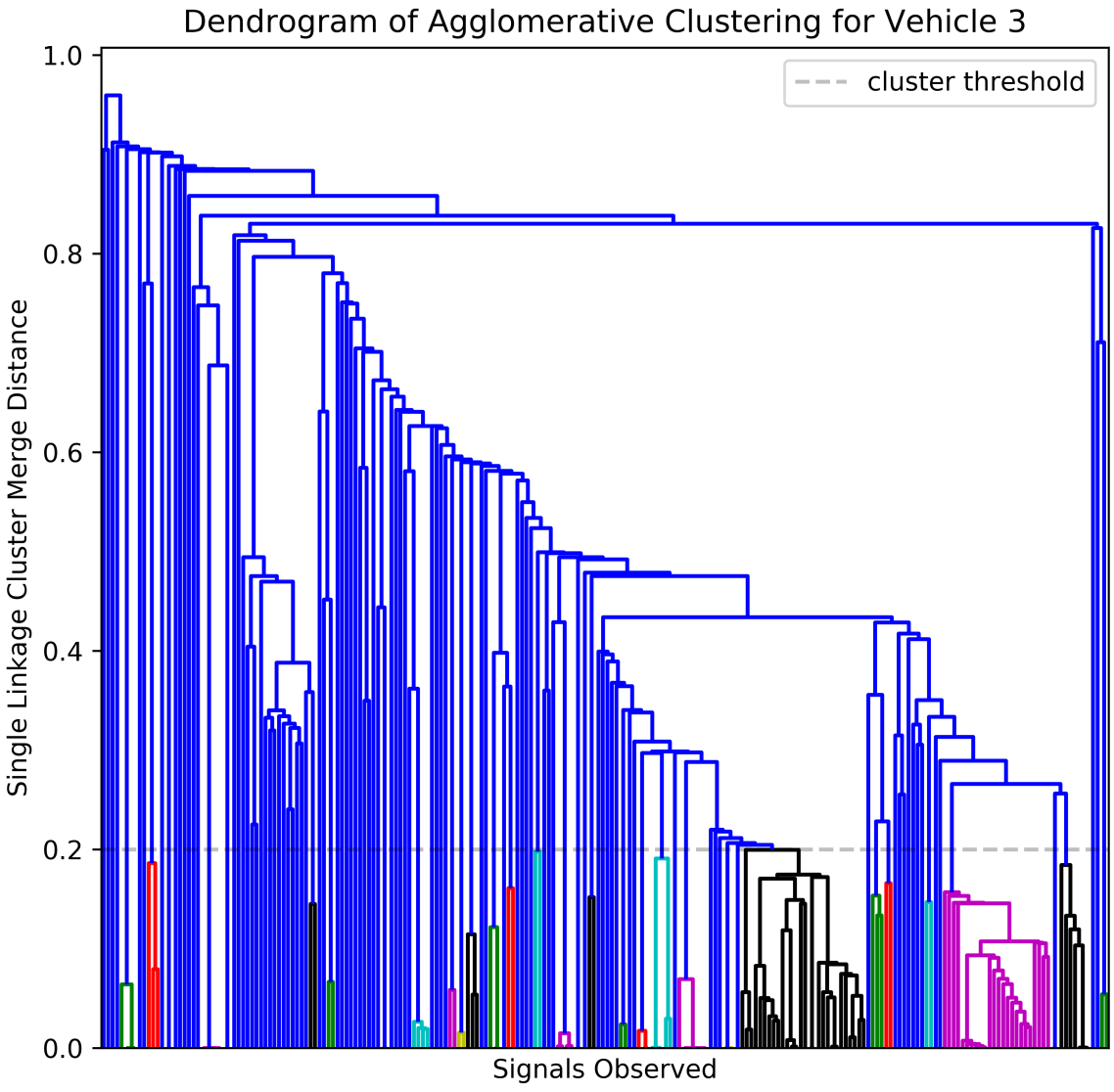


Figure 60. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 3

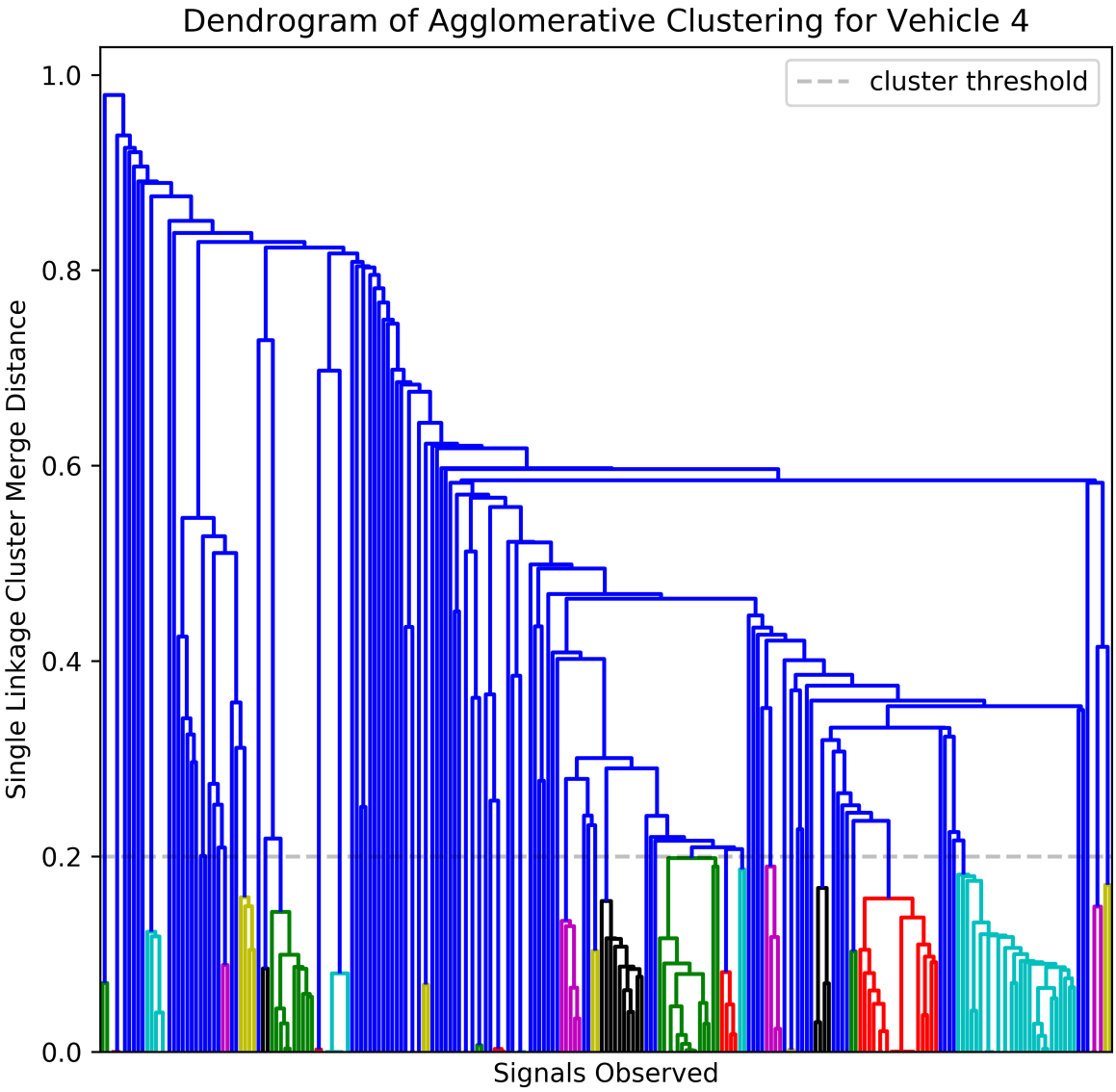


Figure 61. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 4

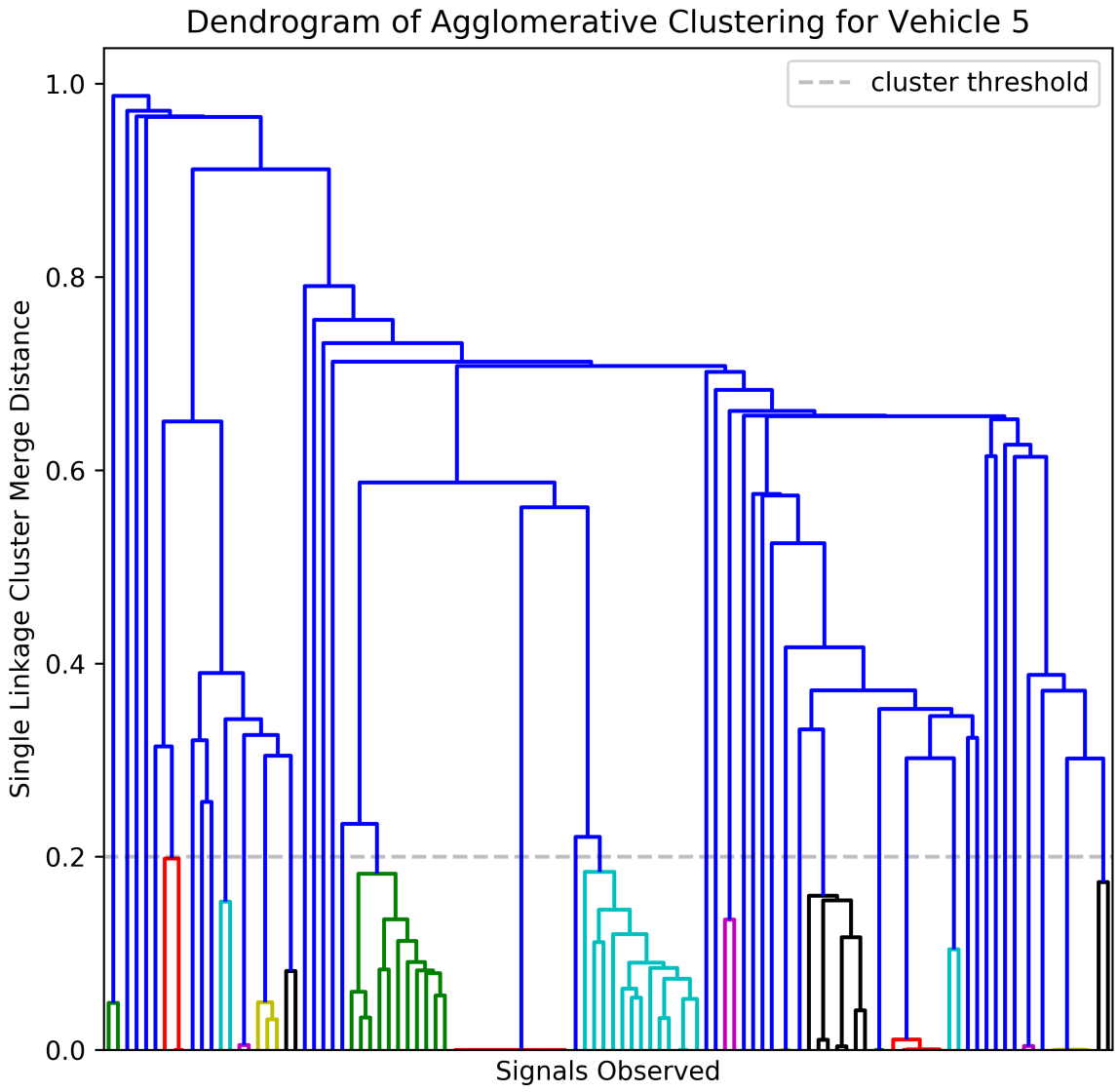


Figure 62. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 5

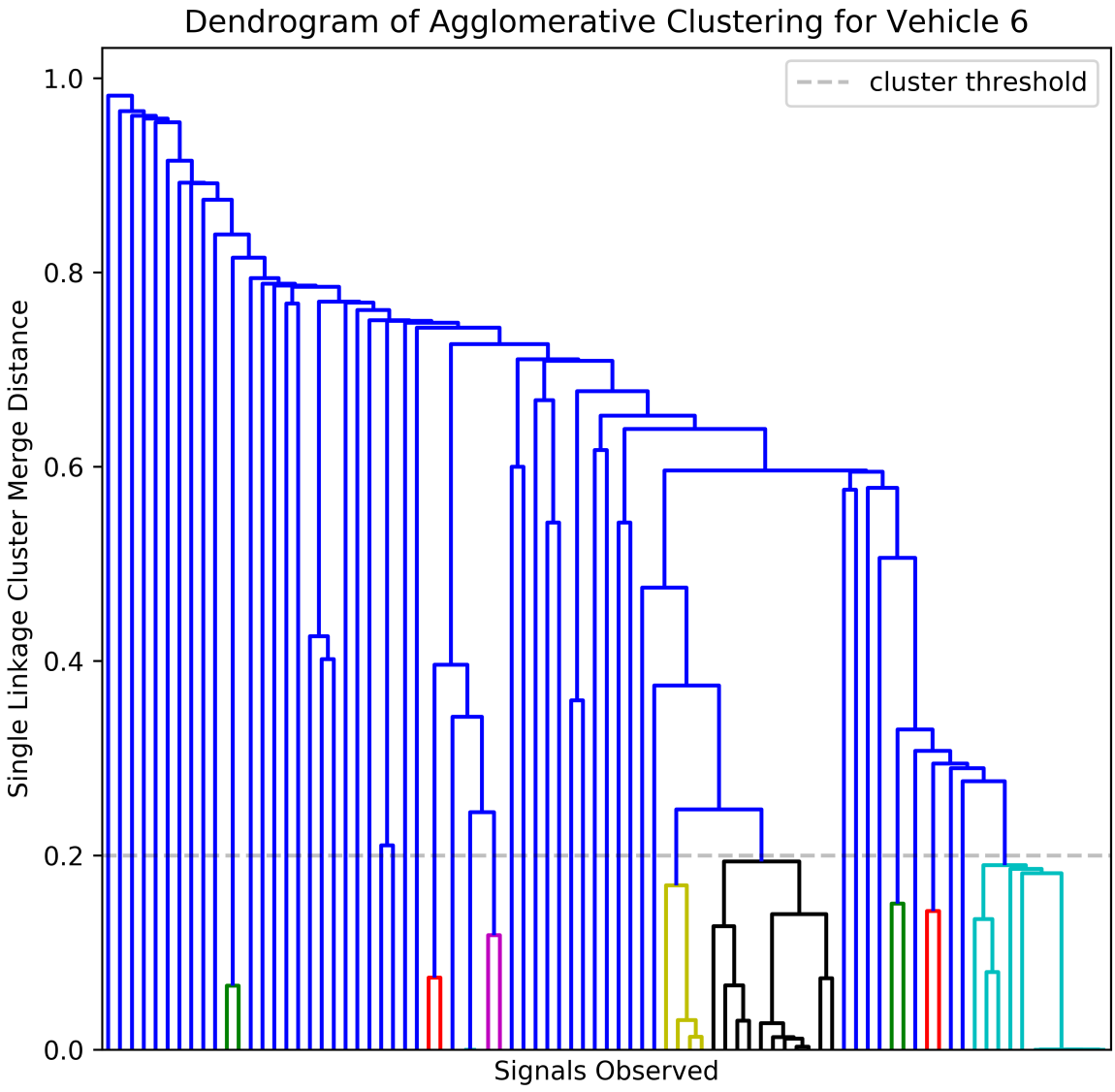


Figure 63. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 6

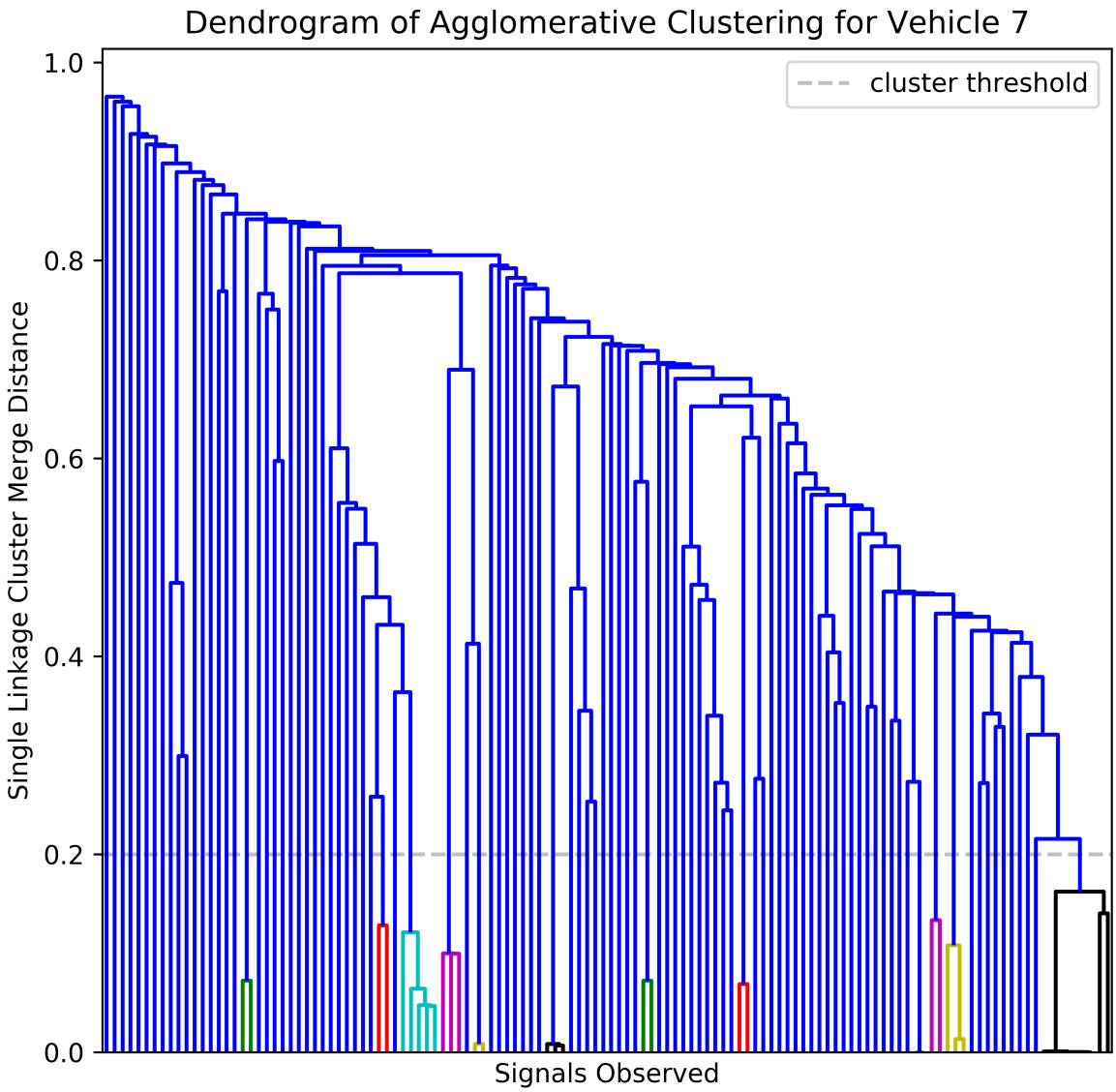


Figure 64. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 7

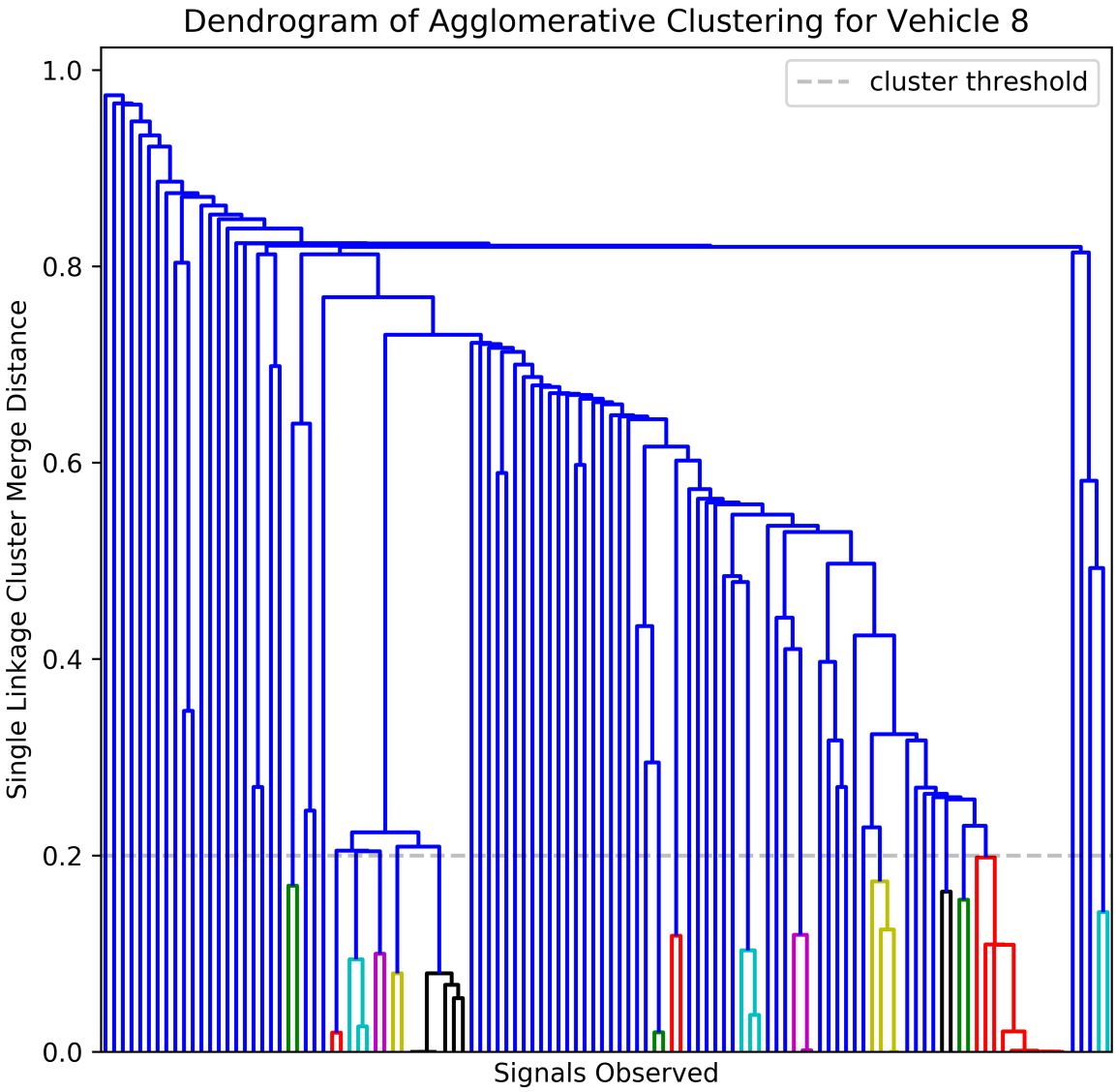


Figure 65. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 8

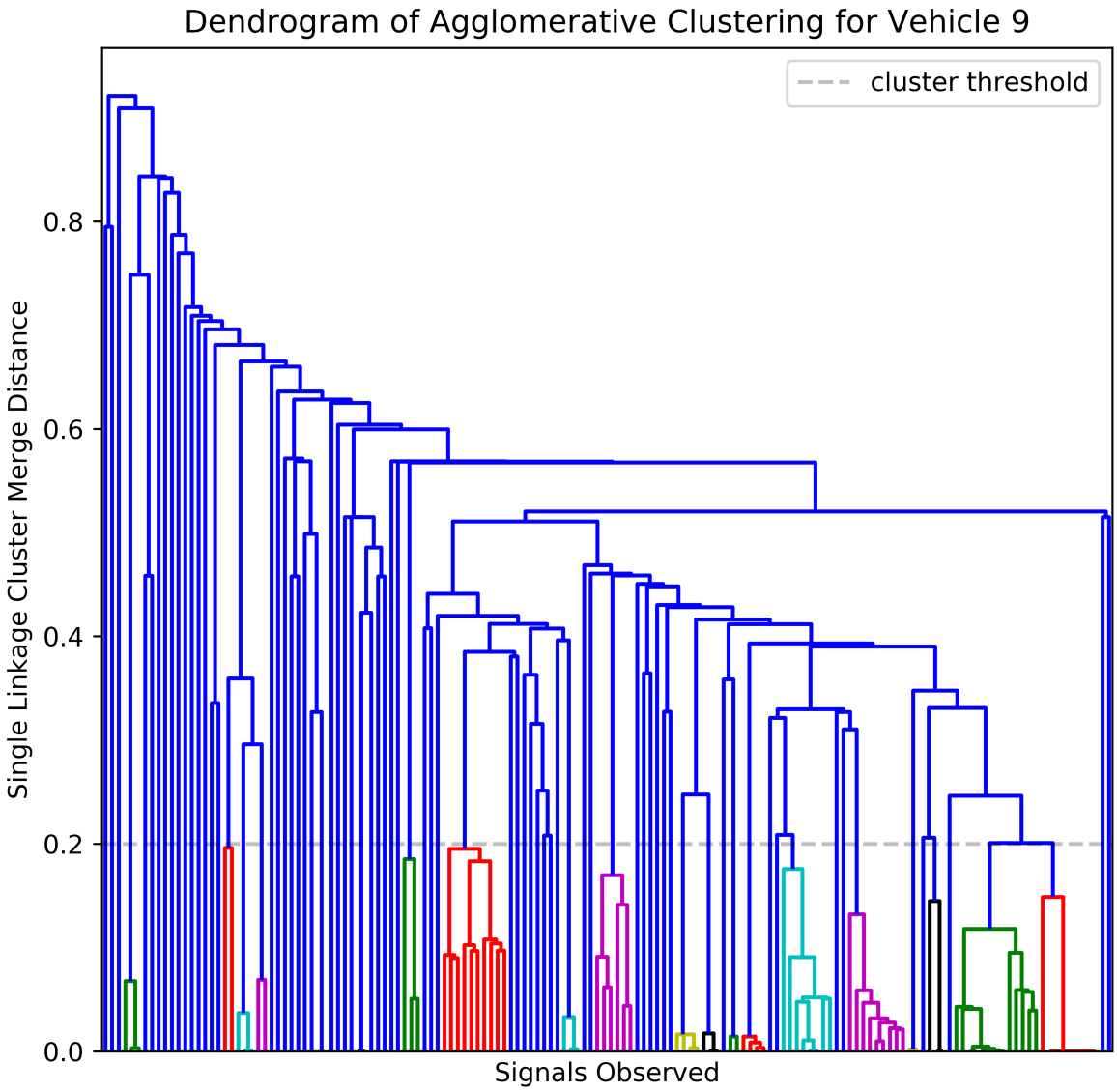


Figure 66. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 9

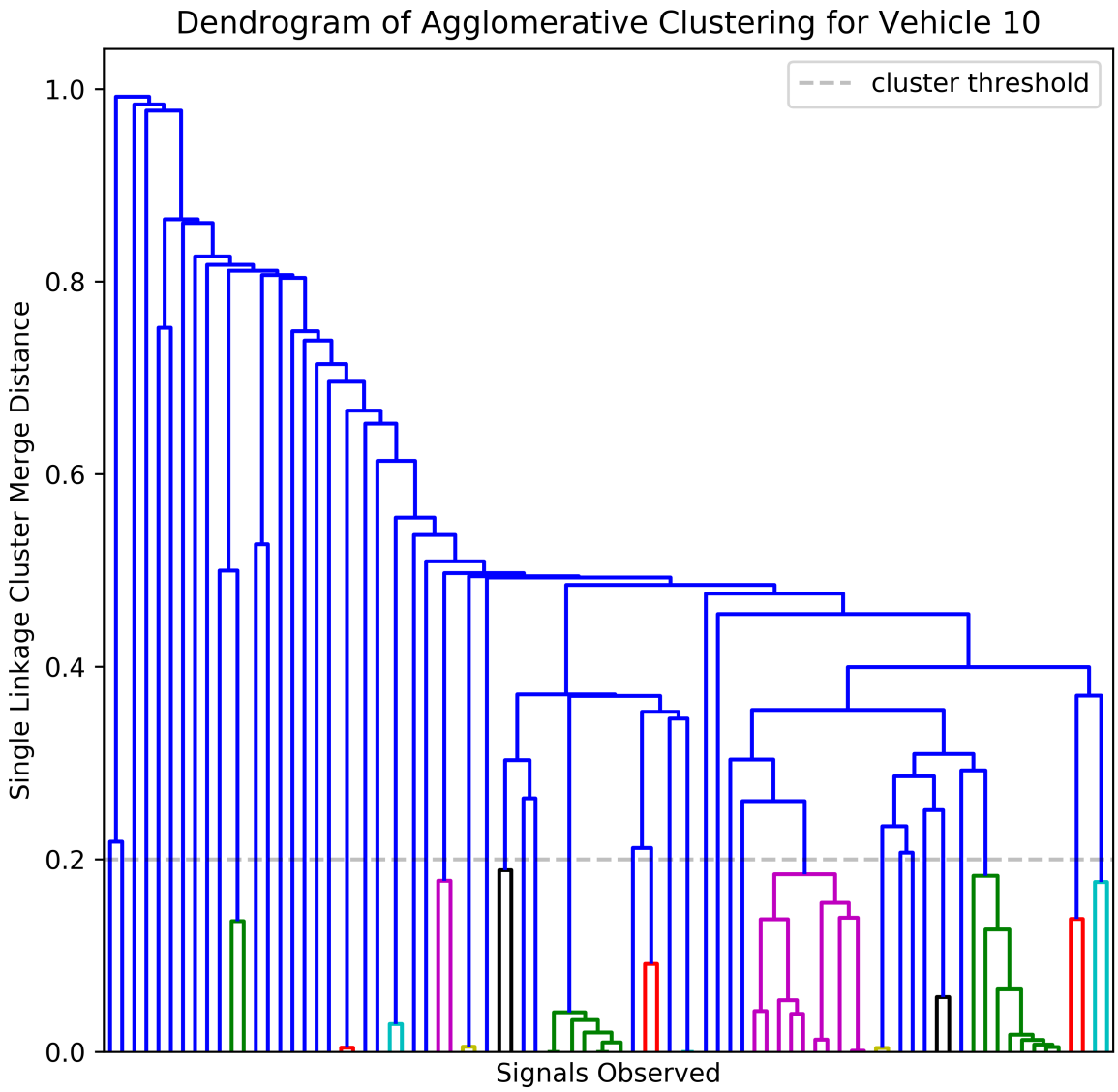


Figure 67. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 10

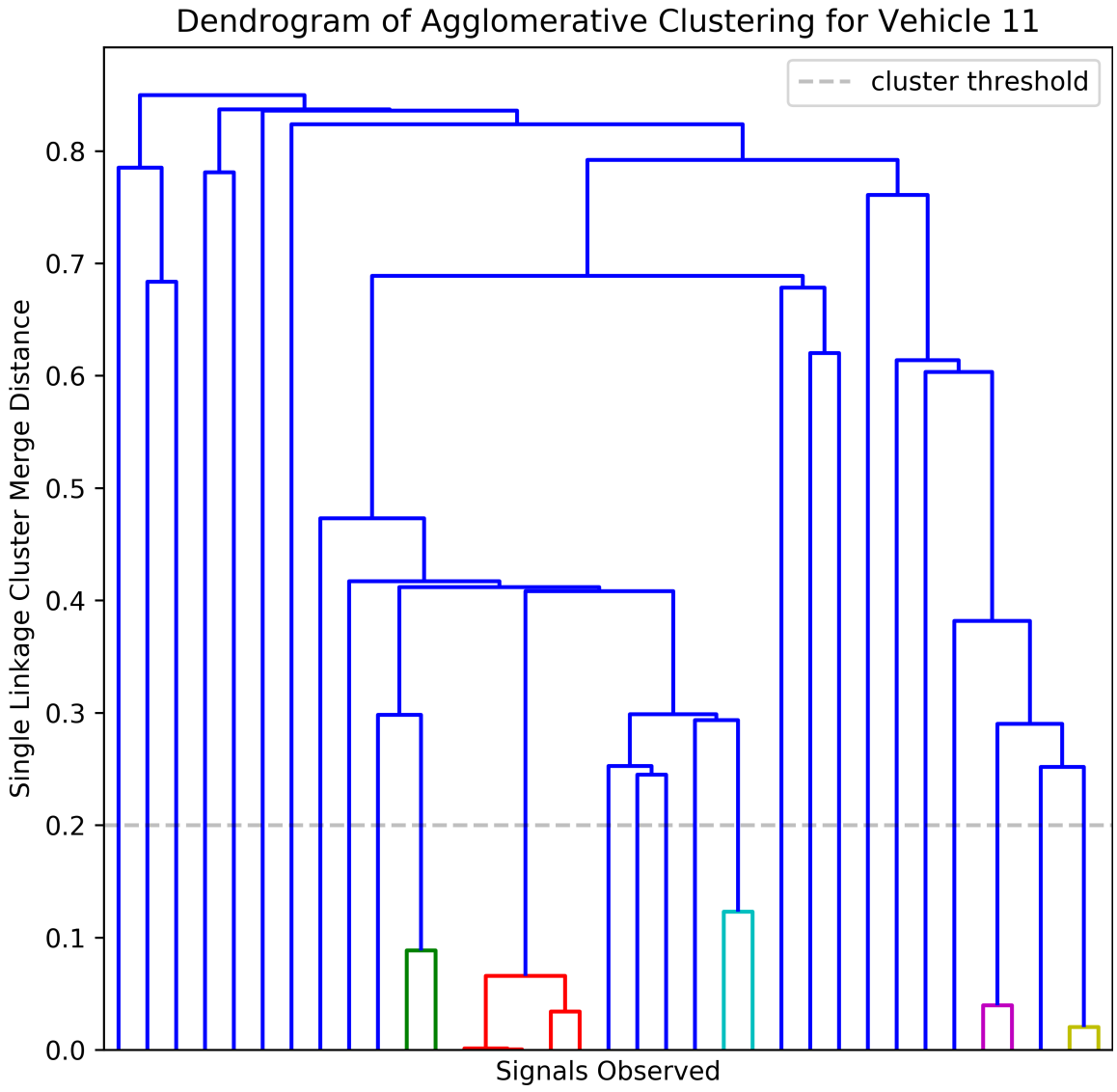


Figure 68. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 11

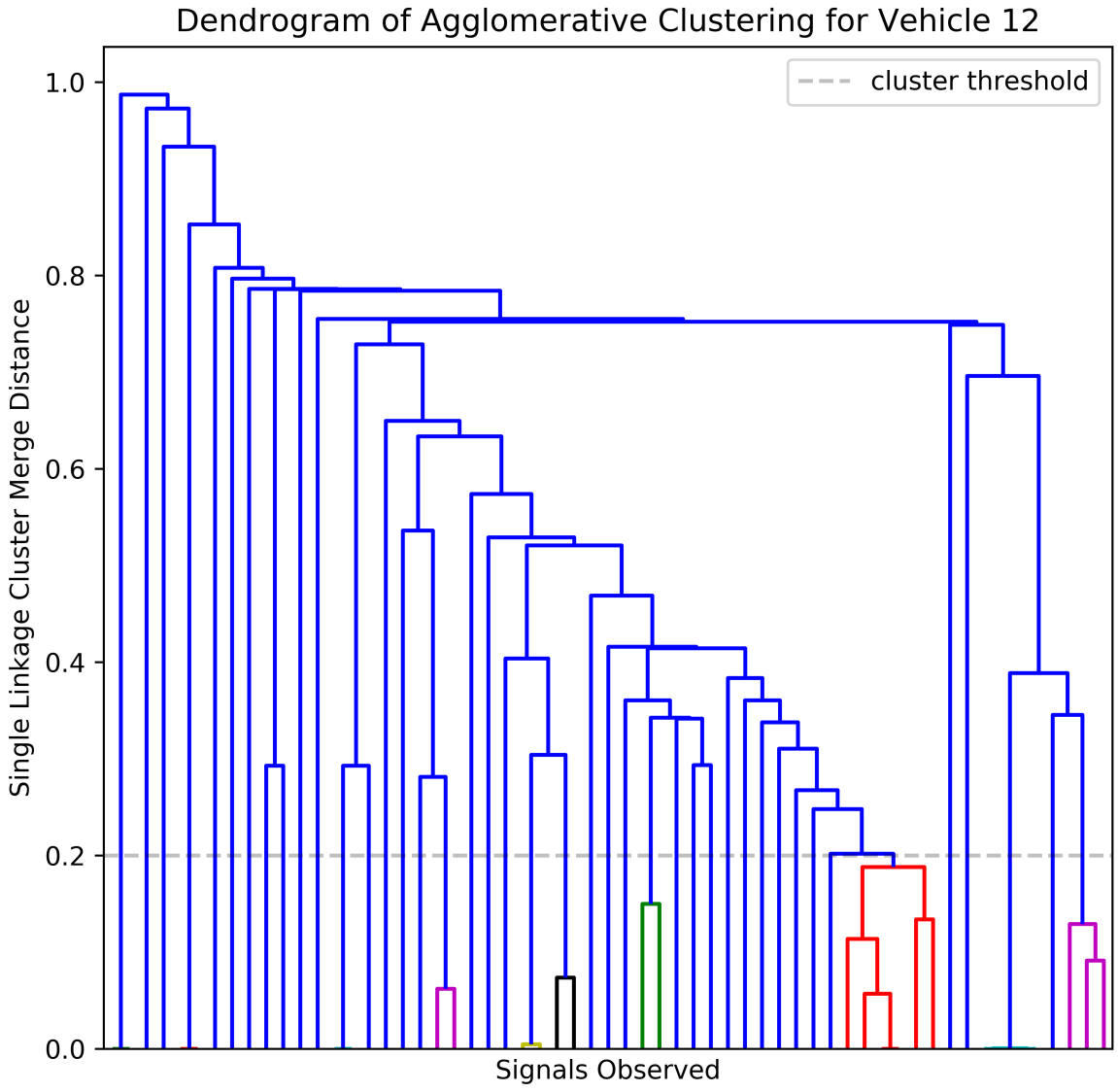


Figure 69. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 12

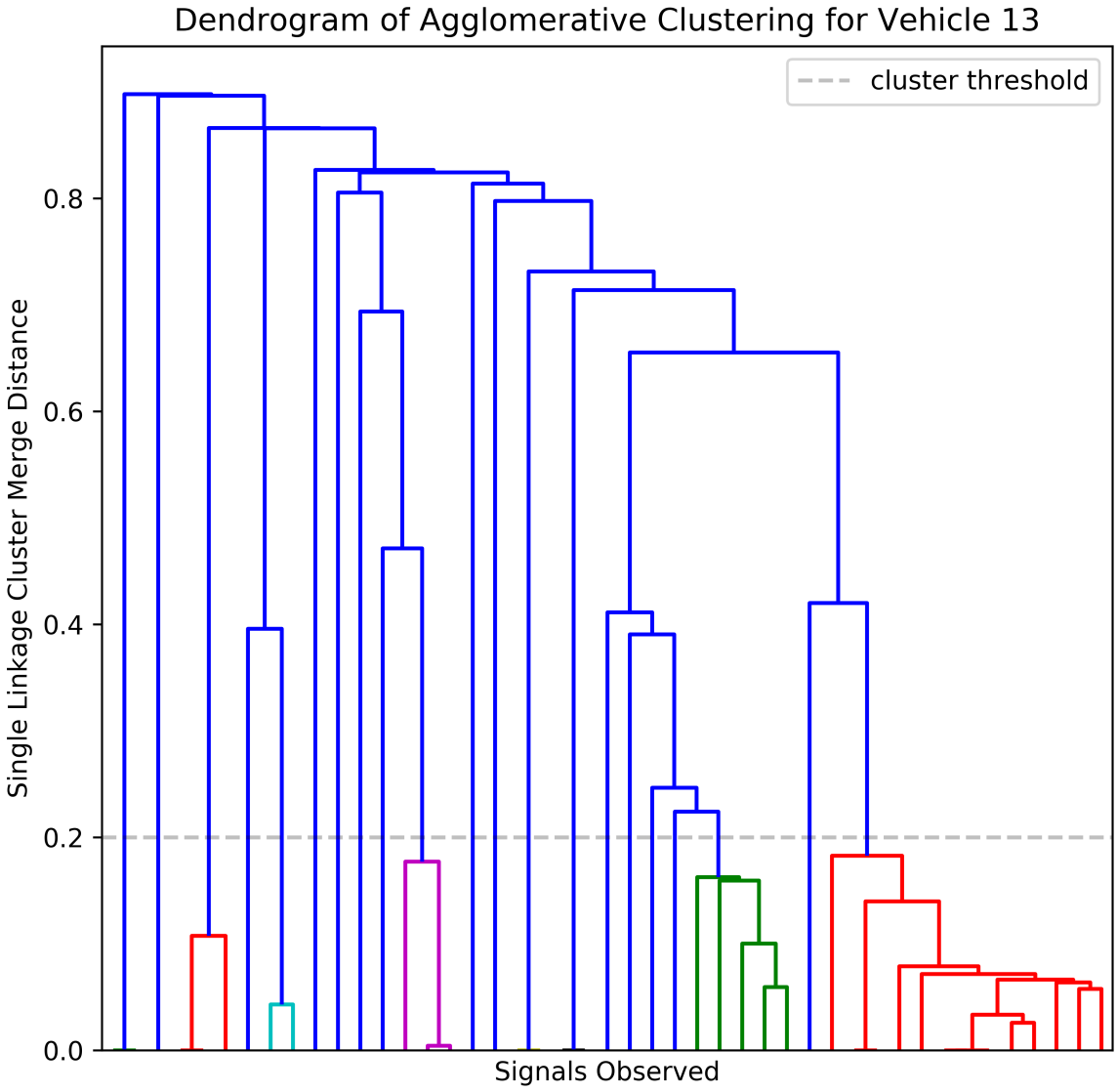


Figure 70. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 13

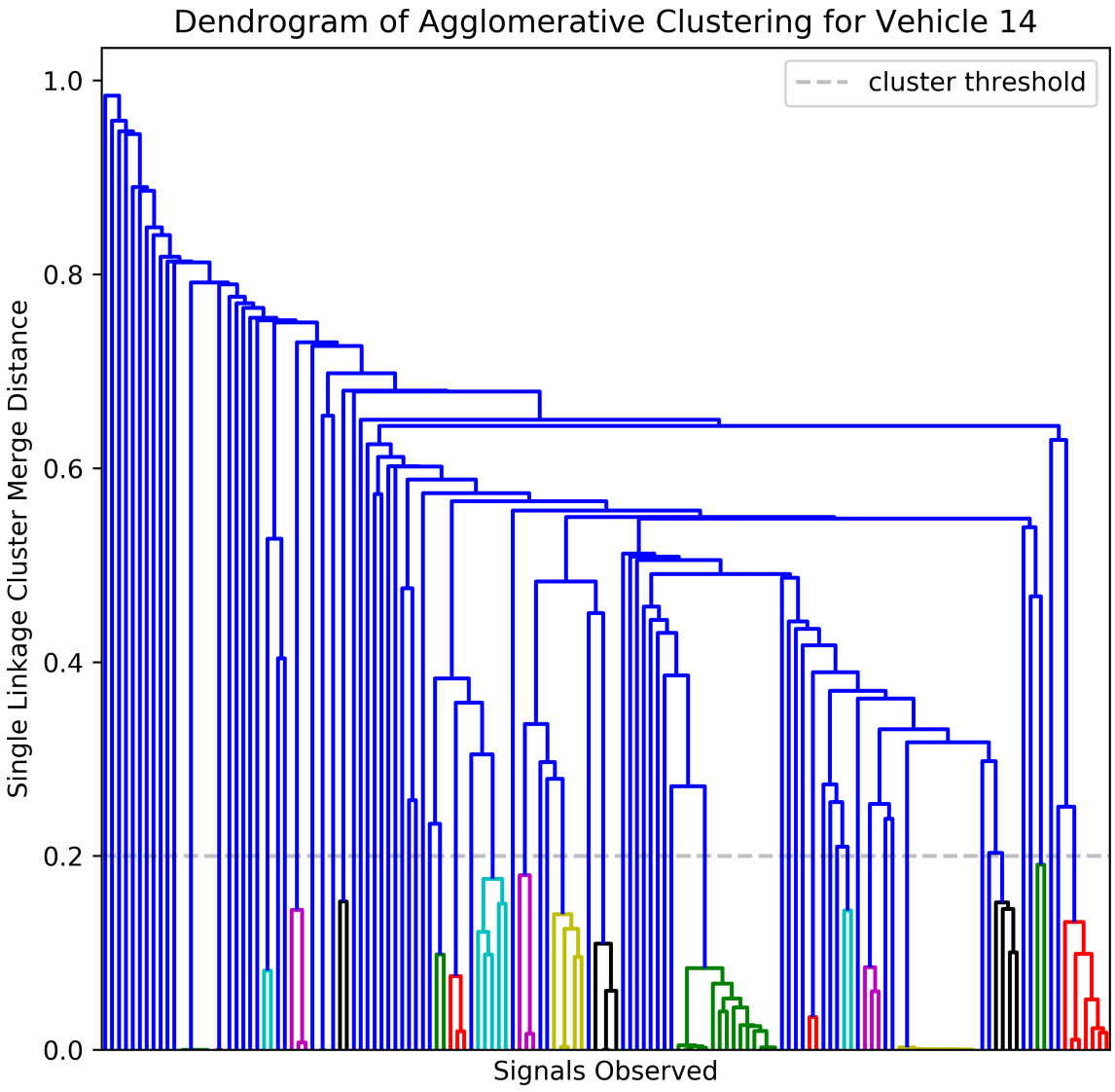


Figure 71. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 14

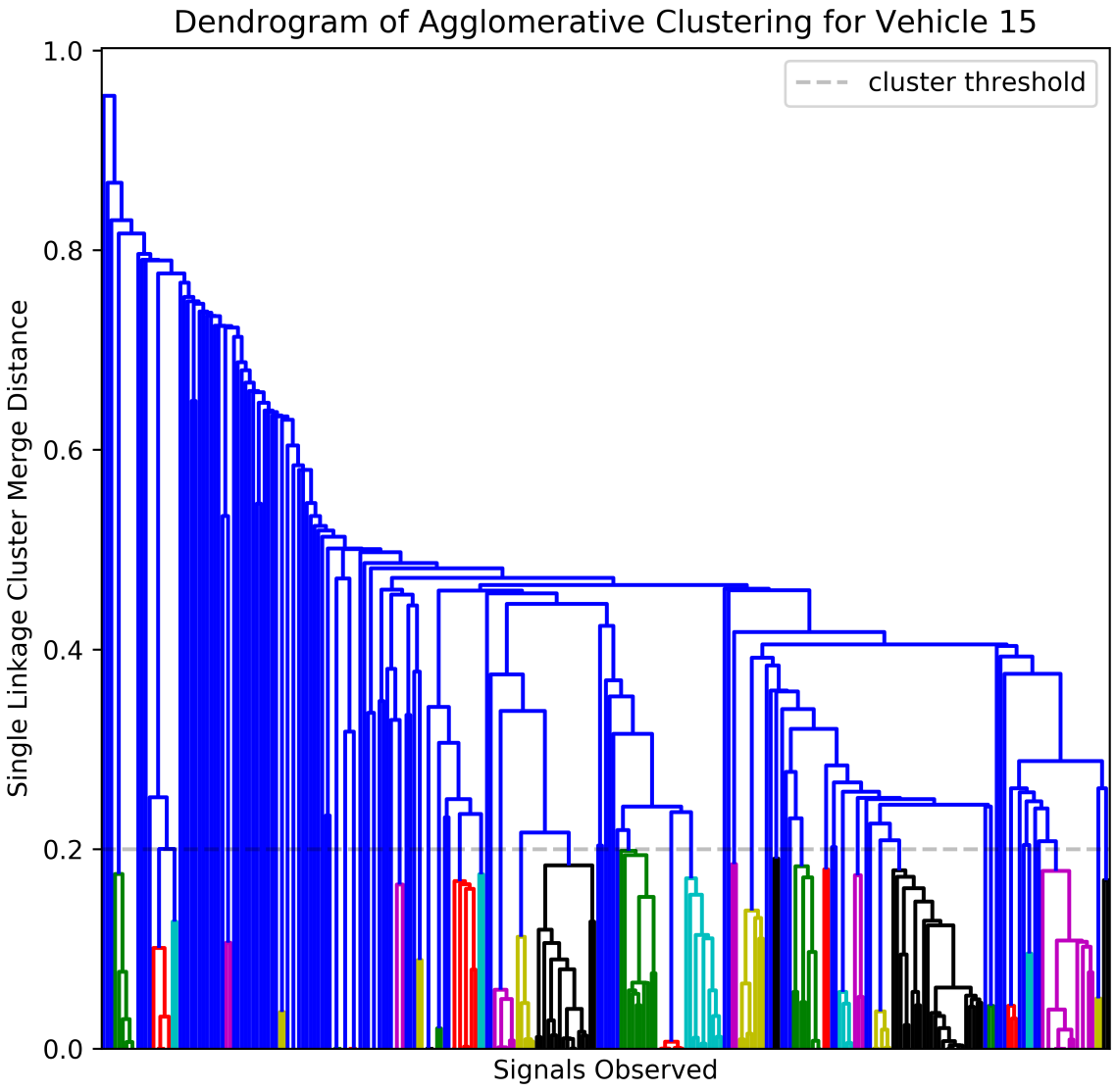


Figure 72. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 15

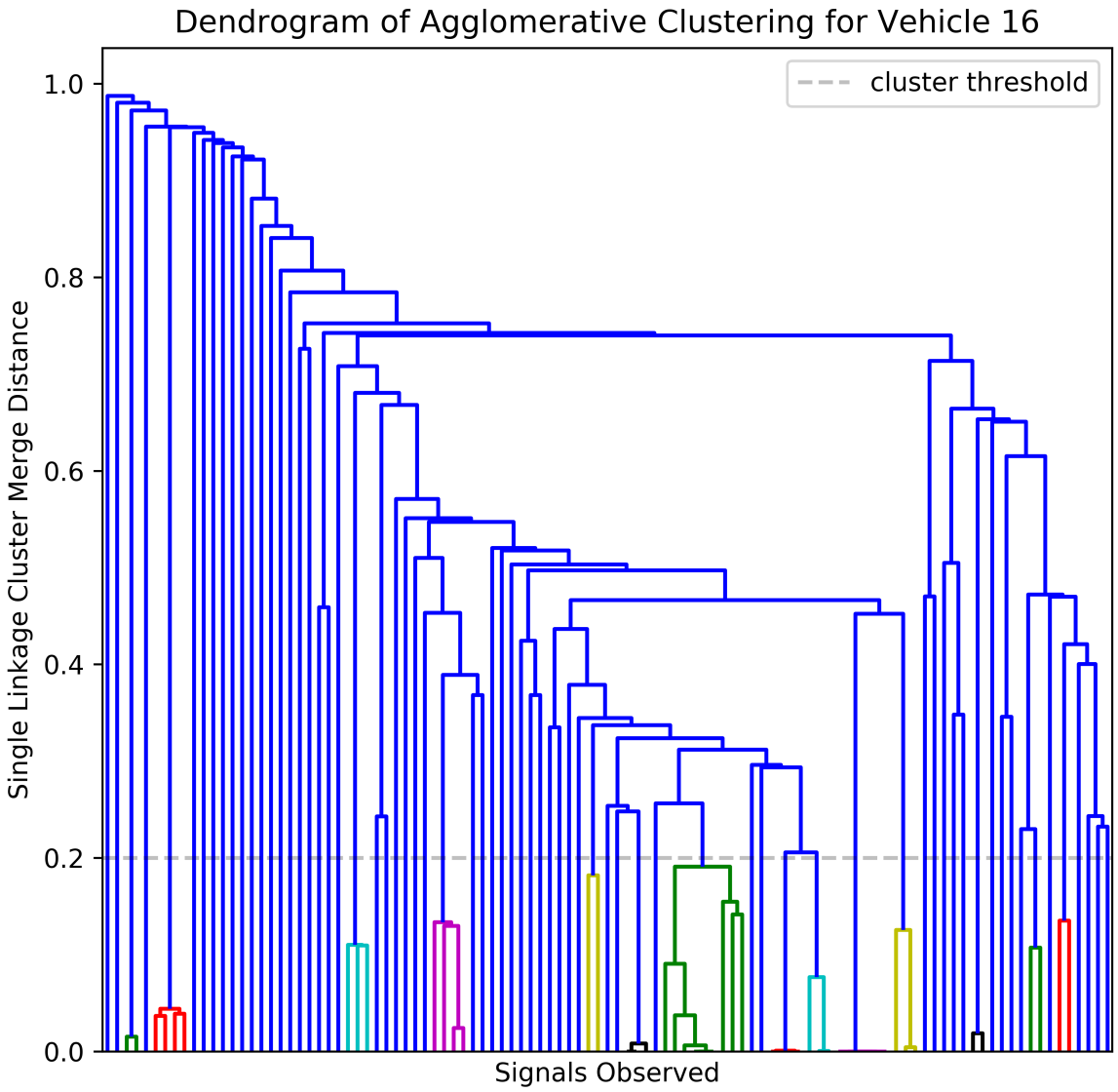


Figure 73. Dendrogram of Agglomerative Hierarchical Clustering for Vehicle 16

Bibliography

1. R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 2nd ed. OTexts, 2018.

2. ISO/TC 22/SC 31, "ISO 11898-1:2015: Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signaling," ISO 11898-1:2015. p. 65, 2015.

3. F. Hartwich, F. Hartwich, R. B. Gmbh, and D.- Reutlingen, "CAN with flexible data-rate," CAN Newsletter, Reutlingen, pp. 10–19, Feb-2012.

4. Ixia, "Automotive Ethernet: An Overview," Ixia, Calabasas, CA, 915–3510–01 Rev. A, 2014. [Online]. Available: https://support.ixiacom.com/sites/default/files/resources/whitepaper/ixia-automotive-ethernet-primer-whitepaper_1.pdf. [Accessed: 08-Sep-2017]

5. Mopar and FCA, "FCA wiTECH Tools Aftermarket Sales and Support," 2017. [Online]. Available: https://aetools.witechtools.com/. [Accessed: 05-Sep-2017].

6. OBDII365.com, "FCA wiTech MicroPod 2 Diagnostic Programming Tool V17.03.01 for Chrysler Multi-language," 2017. [Online]. Available: http://www.obdii365.com/wholesale/witech-micropod-2-diagnostic-programming-tool.html. [Accessed: 05-Sep-2017].

7. Honda, "Honda and Acura Vehicle Diagnostics," 2017. [Online]. Available: http://techinfo.honda.com/rjanisis/pubs/web/RJAAI001_TOOLS1.htm. [Accessed: 05-Sep-2017].

8. M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown CAN bus networks," Vehicular Communications, vol. 9, pp. 43–52, 2017.

9. T. Glennan, C. Leckie, and S. M. Erfani, "Improved Classification of Known and Unknown Network Traffic Flows Using Semi-supervised Machine Learning," in 21st Australasian Conference on Information Security and Privacy (ACISP), 2016, vol. 2, pp. 493–501.

10. J. Zhang, C. Chen, Y. Xiang, and W. Zhou, "Semi-supervised and compound classification of network traffic," in Proceedings 32nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), 2012, pp. 617–621.

11. G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, 6th ed. New York, New York, USA: Springer Science+Business Media, 2013.

12. Sugihara Lab, "Empirical Data Modeling: Quantitative Ecology and Data-Driven Theory," University of California at San Diego, 2018. [Online]. Available: http://deepeco.ucsd.edu/nonlinear-dynamics-research/edm/. [Accessed: 08-Aug-2018].

13. G. Sugihara et al., "Detecting causality in complex ecosystems," Science, vol. 338, no. 6106, pp. 496–500, 2012.

14. O. Balci, "Verification validation and accreditation of simulation models," in Proceedings of the 29th Conference on Winter Simulation, 1997, pp. 135–141.

15. L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," IEEE Transactions on Biomedical Engineering, vol. 43, no. 4, pp. 348–356, 1996.

16. S. Salvador and P. Chan, "FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space," Intelligent Data Analysis, vol. 11, pp. 561–580, 2007.

17. C. Anderson and G. Sugihara, "Simplex Projection," Simplex Portal, 2018. [Online]. Available: http://deepeco.ucsd.edu/simplex/. [Accessed: 06-Sep-2018].

18. W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic Protocol Reverse Engineering from Network Traces," in USENIX Security, 2007, no. 2, pp. 199–212.

19. Y. Wang, Y. Xiang, J. Zhang, and S. Yu, "A novel semi-supervised approach for network traffic clustering," in 5th International Conference on Network and System Security (NSS), 2011, pp. 169–175.

20. SAE International, "SAE J1979: E/E Diagnostic Test Modes," SAE, 2017. [Online]. Available: https://www.sae.org/standards/content/j1979_201202/.

21. C. Valasek and C. Miller, "Adventures in Automotive Networks and Control Units," IO Active, 2013. [Online]. Available: http://www.ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf. [Accessed: 02-Jul-2018].

22. K. Koscher et al., "Experimental Security Analysis of a Modern Automobile," in 2010 IEEE Symposium on Security and Privacy, 2010, p. 16.

23. E. Weise, "Chinese group hacks a Tesla for the second year in a row," USA Today, 2017. [Online]. Available: https://www.usatoday.com/story/tech/2017/07/28/chinese-group-hacks-tesla-second-year-row/518430001/. [Accessed: 16-Sep-2017].

24. R. Hotten, "Volkswagen: The scandal explained," BBC News, p. 1, 10-Dec-2015. [Online]. Available: http://www.bbc.com/news/business-34324772. [Accessed: 21-Jul-2018].

25. B. Sorokanich, "The Facts Behind Every Major Automaker Emissions Cheating Scandal Since VW," roadandtrack.com, 25-May-2016. [Online]. Available: https://www.roadandtrack.com/new-cars/car-technology/a29293/vehicle-emissions-testing-scandal-cheating/. [Accessed: 21-Jul-2018].

26. P. Lyon, "Nissan Admits To Falsifying Emissions Tests," Forbes, p. 2, 09-Jul-2018. [Online]. Available: https://www.forbes.com/sites/peterlyon/2018/07/09/nissan-admits-to-falsifying-emissions-tests/#54dcbfa6376c. [Accessed: 21-Jul-2018].

27. J. Sametinger, J. Rozenblit, R. Lysecky, and P. Ott, "Security challenges for medical devices," Communications of the ACM, vol. 58, no. 4, pp. 74–82, 2015.

28. National Institute of Standards and Technology (NIST), "Process or Product Monitoring and Control: Introduction to Time Series Analysis," in NIST/SEMATECH e-Handbook of Statistical Methods, 1st ed., Gaithersburg, MD, USA: NIST, 2012, p. 281.

29. A. V. Aho and J. D. Ullman, "The Theory of Parsing, Translation, and Compiling," Prentice-Hall Series in Automatic Computation, vol. 53, no. 9, pp. 1689–1699, 2013.

30. Y. Yaari, "Segmentation of Expository Texts by Hierarchical Agglomerative Clustering," CoRR, vol. cmp-lg/970, p. 7, Sep. 1997.

31. P. Simon, S. Hsieh, and L. Pr, "Rethinking Chinese Word Segmentation : Tokenization , Character Classification , or Wordbreak Identification," Computational Linguistics, vol. 1, no. June, pp. 69–72, 2007.

32. N. Chomsky, "Three models for the description of language," IEEE Transactions on Information Theory, vol. 2, no. 3, pp. 113–124, 1956.

33. N. Chomsky, "On certain formal properties of grammars," Information and Control, vol. 2, no. 2, pp. 137–167, 1959.

34. C. D. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, 2nd ed. Cambridge: The MIT Press, 1999.

35. S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python, 1st ed., vol. 53, no. 9. Sebastopol: O'Reilly Media, 2009.

36. S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," Journal of Molecular Biology, vol. 48, no. 3, pp. 443–453, 1970.

37. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," Journal of Molecular Biology, vol. 215, no. 3, pp. 403–410, 1990.

38. M. Daszykowski and B. Walczak, "Density-Based Clustering Methods," Comprehensive Chemometrics, vol. 2, pp. 635–654, 2010.

39. scikit-learn, "sklearn.cluster.AgglomerativeClustering," Scikit-learn, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering. [Accessed: 01-Jan-2017].

40. M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile Driver Fingerprinting," Proceedings on Privacy Enhancing Technologies, vol. 2016, no. 1, pp. 34–51, 2016.

41. A. D. Atkinson, R. R. Hill, J. J. Pignatiello, G. G. Vining, E. D. White, and E. Chicken, "Wavelet ANOVA approach to model validation," Simulation Modeling Practice and Theory, vol. 78, pp. 18–27, 2017.

42. C. Torrence and G. P. Compo, "A Practical Guide to Wavelet Analysis," Bulletin of the American Meteorological Society, vol. 79, no. 1, pp. 61–78, 1998.

43. B. D. Fulcher, "Feature-based time-series analysis," in Feature Engineering for Machine Learning and Data Analytics, 2018, pp. 87–116.

44. R. Prado and M. West, Time series: Modeling, Computation, and Inference, 1st ed. Boca Raton: Chapman & Hall/CRC, 2011.

45. T. Kuwahara et al., "Supervised and Unsupervised Intrusion Detection Based on CAN Message Frequencies for In-vehicle Network," Journal of Information Processing, vol. 26, no. 0, pp. 306–313, 2018.

46. R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification, 2nd ed. New York: John Wiley & Sons, 2001.

47. R. M. O'Keefe and D. E. O'Leary, "Expert system verification and validation: a survey and tutorial," Artificial Intelligence Review, vol. 7, no. 1, pp. 3–42, 1993.

48. I. F. Spellerberg and P. J. Fedor, "A tribute to Claude-Shannon (1916-2001) and a plea for more rigorous use of species richness, species diversity and the 'Shannon-Wiener' Index," Global Ecology and Biogeography, vol. 12, no. 3, pp. 177–179, 2003.

49. K. Choi, Y. Son, J. Noh, H. Shin, J. Choi, and Y. Kim, "Dissecting Customized Protocols: Automatic Analysis for Customized Protocols Based on IEEE 802.15.4," in 9th International Conference on Security of Information and Networks, 2016, pp. 183–193.

50. R. L. S. Puuperä, "Domain Model Based Black Box Fuzzing Using Regular Languages," University of Oulu, 2010.

51. J. Röning, "PROTOS Protocol Genome Project," Oulu University Secure Programming Group, 2010. [Online]. Available: https://www.ee.oulu.fi/roles/ouspg/genome. [Accessed: 01-Jan-2017].

52. Y. Wang et al., "A semantics aware approach to automated reverse engineering unknown protocols," in 20th IEEE International Conference on Network Protocols (ICNP), 2012, pp. 1–10.

53. C. W. Chang, M. Ushio, and C. H. Hsieh, "Empirical dynamic modeling for beginners," Ecological Research, vol. 32, no. 6, pp. 785–796, 2017.

54. ACM TCPS, "Cyber-Physical Systems (TCPS): About," ACM Transactions on Cyber-Physical Systems (TCPS), 2018. [Online]. Available: https://tcps.acm.org/about.cfm. [Accessed: 01-Oct-2018].

55. R. Bosch, "BOSCH CAN Specification," Bosch, 1991. [Online]. Available: http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf. [Accessed: 01-Jan-2017].

56. International Organization for Standardization, "ISO Standard 11898-1:2003(E) Controller Area Network (CAN)," 2003.

57. M. Farsi, K. Ratcliff, and M. Barbosa, "An overview of controller area network," Computing & Control Engineering Journal, vol. 10, no. 3, pp. 113–120, 1999.

58. C. Watterson, "Controller Area Network (CAN) Implementation Guide," Norwood, MA, AN-1123, 2012. [Online]. Available: http://www.analog.com/media/en/technical-documentation/application-notes/an-1123.pdf. [Accessed: 10-Dec-2016].

59. National Instruments, "Controller Area Network (CAN) Overview," ni.com, 2014. [Online]. Available: http://www.ni.com/white-paper/2732/en/. [Accessed: 10-Jul-2018].

60. S. Corrigan, "Introduction to the Controller Area Network (CAN)," 2016. [Online]. Available: http://www.ti.com/lit/an/sloa101b/sloa101b.pdf. [Accessed: 10-Dec-2016].

61. National Instruments, "Understanding CAN with Flexible Data-Rate (CAN FD) Motivation for CAN FD NI Tools for CAN FD Communication," ni.com, 2018. [Online]. Available: http://www.ni.com/white-paper/52288/en/. [Accessed: 15-Aug-2018].

62. Federal Register, "Environmental Protection Agency 40 CFR Part 86," Federal Register, vol. 70, no. 243. pp. 25–33, 2005.

63. SAE International, "SAE J1939: Application Layer - Diagnostics," SAE, 2017. [Online]. Available: https://www.sae.org/standardsdev/groundvehicle/j1939a.htm. [Accessed: 10-Dec-2016].

64. T. Montes, "Heavy duty OBD update," in SAE 2015 On-Board Diagnostic Symposium, 2015, pp. 1–36.

65. ISO, "ISO 27145-1:2012(en), Road vehicles — Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD) communication requirements — Part 1: General information and use case definition," 2012. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:27145:-1:ed-1:v1:en. [Accessed: 08-Sep-2017].

66. T. Hogenmüller, "Call For Interest (CFI): Automotive Ethernet 1 Twisted Pair 100 Mbit/s Ethernet (1TPCE)," in Call for Interest at IEEE 802.3 Working Group, 2014, pp. 1–41.

67. ISO/TC 22/SC 31, "ISO 15765-2:2004: Road Vehicles - Diagnostics on Controller Area Networks (CAN) - Network Layer Services," 2004.

68. ISO/TC 22/SC 31, "ISO 15031-6:2015: Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 6: Diagnostic trouble code definitions," ISO, 2015. [Online]. Available: https://www.iso.org/standard/66369.html. [Accessed: 01-Jan-2017].

69. BMW and Mini, "BMW/Mini: Online Service System-OSS Installation and User Guide," BMW Tech Info, 2017. [Online]. Available: https://www.bmwtechinfo.com/tiscode/workshop/oss_user_guide.pdf. [Accessed: 23-Aug-2017].

70. ISO/TC 22/SC 31, "ISO 15765-4: Road vehicles — Diagnostics on Controller Area Networks (CAN) - Requirements for emissions-related systems." pp. 1–28, 2010.

71. S. S. Skiena, Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, 1st ed. Michigan: Addison-Wesley, 1990.

72. C. Valasek and C. Miller, "Automotive Hacking," in Hackers to Hackers (H2HC) Conference, 2014, p. 1. [Online]. Available: https://www.youtube.com/watch?v=q6LYZ91u2gg. [Accessed: 06-Sep-2018].

73. A. Greenberg, "Hackers remotely kill a Jeep on the highway—with me in it," Wired.com, 21-Jul-2015. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. [Accessed: 01-Jul-2016].

74. National Highway Traffic Safety Administration, "Report No. DOT HS 812 333: Cybersecurity best practices for modern vehicles," Washington DC, 2016.

75. R. Triebel, M. Krucker, and R. Bitzi, "Communication with a Toyota Prius: Decoding of the CAN Protocol and Development of a Graphical Interface," ETH Zurich Autonomous System Lab (ASL), 2009. [Online]. Available: https://attachments.priuschat.com/attachment-files/2017/04/122809_Communication_with_a_Toyota_Prius.pdf. [Accessed: 01-Dec-2016].

76. A. Chattopadhyay, A. Prakash, and M. Shafique, "Secure Cyber-Physical Systems: Current trends, tools and open research problems," in Design, Automation & Test in Europe Conference & Exhibition, 2017, pp. 1104–1109.

77. P. N. Borazjani, C. E. Everett, and D. Mccoy, "OCTANE: An Extensible Open Source Car Security Testbed," in Embedded Security In Cars (ESCAR), 2014, pp. 1–10.

78. F. Alam, R. Mehmood, I. Katib, and A. Albeshri, "Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT)," in International Workshop on Data Mining in IoT Systems (DaMIS 2016), 2016, vol. 98, no. 1, pp. 437–442.

79. Scikit-learn.org, "Clustering," Scikit-learn, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/clustering.html#clustering. [Accessed: 01-Jan-2017].

80. J. Erman and M. Arlitt, "Traffic classification using clustering algorithms," in 2006 SIGCOMM Workshop on Mining Network Data, 2006, pp. 281–286.

81. F. Takens, "Detecting strange attractors in turbulence," Dynamical Systems and Turbulence, pp. 366–381, 1981.

82. A. T. Clark et al., "Spatial convergent cross mapping to detect causal relationships from short time series," Ecology, vol. 96, no. 5, pp. 1174–1181, 2015.

83. G. Sugihara and R. M. May, "Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series," Nature, vol. 344, pp. 734–741, 1990.

84. G. Sugihara, "Nonlinear Forecasting for the Classification of Natural Time Series," Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences, vol. 348, no. 1688, pp. 477–495, 1994.

85. H. Ye, A. Clark, E. Deyle, and G. Sugihara, "rEDM: an R package for Empirical Dynamic Modeling and Convergent Cross-Mapping," r-project.org, 2018. [Online]. Available: https://cran.r-project.org/web/packages/rEDM/vignettes/rEDM-tutorial.html. [Accessed: 01-Aug-2018].

86. J. M. Lee, Introduction to Topological Manifolds, 1st ed. New York: Springer, 2000.

87. H. Whitney, "Differentiable Manifolds in Euclidean Space," Proceedings of the National Academy of Sciences, vol. 21, no. 7, pp. 462–464, Jul. 1935.

88. C. W. J. Granger, "Investigating Causal Relations by Econometric Models and Cross-spectral Methods," Econometrica, vol. 37, no. 3, p. 424, 1969.

89. H. Ye et al., "Package 'rEDM': Applications of Empirical Dynamic Modeling from Time Series," GitHub.com, 2018. [Online]. Available: https://github.com/ha0ye/rEDM.

90. A. D. Atkinson, "Wavelet-Based Simulation Model Validation of Functional Data," Air Force Institute of Technology, 2017.

91. R. G. Sargent, "Verification and validation of simulation models," Journal of Simulation, vol. 7, no. 1, pp. 12–24, 2013.

92. H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, no. 1, pp. 43–49, 1978.

93. C. A. Ratanamahatana and E. Keogh, "Three Myths about Dynamic Time Warping Data Mining," Proceedings of the 2005 SIAM International Conference on Data Mining, no. 1, April 2005, pp. 506–510, 2005.

94. Shiroyagi Corporation, "FastDTW," GitHub.com, 2018. [Online]. Available: https://github.com/slaypni/fastdtw.

95. P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," in IEEE Symposium on Security and Privacy, 2009, pp. 110–125.

96. Y. Wang, Z. Zhang, D. Yao, B. Qu, and L. Guo, "Inferring Protocol State Machine from Network Traces: A Probabilistic Approach," in International Conference on Applied Cryptography and Network Security, 2011, pp. 1–18.

97. A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language (EMNLP-CoNLL), 2007, pp. 410–420.

98. E. W. Weisstein, "Symmetric Difference," MathWorld–A Wolfram Web Resource, 2018. [Online]. Available: http://mathworld.wolfram.com/SymmetricDifference.html.

99. Raspberry Pi Foundation, "Raspbian," Wikipedia, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Raspbian#Version_history. [Accessed: 27-Nov-2018].

100. Raspberry Pi Foundation, "Raspberry Pi Zero W," raspberrypi.org, 2018. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/. [Accessed: 27-Nov-2018].

101. T. E. Oliphant, A guide to NumPy, 1st ed. Moscow: Trelgol Publishing, 2006.

102. W. McKinney, "Data Structures for Statistical Computing in Python," in 9th Python in Science Conference, 2010, pp. 51–56.

103. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

104. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

105. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed. New York, NY, USA: Springer, 2001.

106. E. Versi, "'Gold standard' is an appropriate term.," British Medical Journal, vol. 305, no. 6846, p. 187, 1992.

107. A. Taylor, "Anomaly-based detection of malicious activity in in-vehicle networks," University of Ottawa, 2017.

108. C. Smith and D. Blundell, The Car Hacker's Handbook: A Guide for the Penetration Tester. San Francisco: No Starch Press, 2016.

109. C. Valasek and C. Miller, "Remote Exploitation of an Unaltered Passenger Vehicle," IO Active, 2015. [Online]. Available: http://illmatics.com/Remote Car Hacking.pdf. [Accessed: 10-Apr-2017].

110. C. Miller and C. Valasek, "Car Hacking for Poories," in SyScan'14 Singapore, 2014, p. 1. [Online]. Available: https://www.youtube.com/watch?v=0OKUfPX6JXE. [Accessed: 10-Apr-2017].

111. A. Fukushima, M. Kusano, H. Redestig, M. Arita, and K. Saito, "Metabolomic correlation-network modules in Arabidopsis based on a graph-clustering approach," BMC Systems Biology, vol. 5, pp. 1–12, 2011.

112. A. J. Butte and I. S. Kohane, "Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements," in Pacific Symposium on Biocomputing, 1999, vol. 5, no. 1, pp. 415–426.

113. R. Watrigant, M. Bougeret, R. Giroudeau, and J. C. König, "Sum-Max Graph Partitioning Problem," in ISCO: International Symposium on Combinatorial Optimization, 2012, pp. 297–308.

114. R. Watrigant, M. Bougeret, R. Giroudeau, and J. C. König, "On the sum-max graph partitioning problem," Theoretical Computer Science, vol. 540–541, no. 1, pp. 143–155, 2014.

115. S. Fortunato and D. Hric, "Community detection in networks: A user guide," Physics Reports, vol. 659, pp. 1–44, 2016.

116. C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, Jul. 1948.

117. E. W. Weisstein, "Box-and-Whisker Plot," MathWorld–A Wolfram Web Resource, 2018. [Online]. Available: http://mathworld.wolfram.com/Box-and-WhiskerPlot.html. [Accessed: 09-Nov-2018].

118. J. S. Milton and J. C. Arnold, Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences, 4th ed. McGraw-Hill, 2010.

119. SciPy.org, "Hierarchical Clustering," scipy.org, 2018. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html#module-scipy.cluster.hierarchy. [Accessed: 01-Oct-2018].

120. H. Ye, E. R. Deyle, L. J. Gilarranz, and G. Sugihara, "Distinguishing time-delayed causal interactions using convergent cross mapping," Scientific Reports, vol. 5, pp. 1–9, 2015.

121. J. Renze, T. Rowland, and E. W. Weisstein, "Compact Manifold," MathWorld–A Wolfram Web Resourc, 2018. [Online]. Available: http://mathworld.wolfram.com/CompactManifold.html. [Accessed: 26-Nov-2018].

122. Wired, "Genome Model Applied to Software," Wired.com, Nov-2004. [Online]. Available: https://www.wired.com/2004/10/genome-model-applied-to-software/. [Accessed: 05-Jul-2018].

123. T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," Journal of Molecular Biology, vol. 147, no. 1, pp. 195–197, 1981.

124. M. A Beddoe, "Network protocol analysis using bioinformatics algorithms," McAfee, Santa Clara, CA, USA, 1, 2004.

125. J. Antunes, N. Neves, and P. Verissimo, "Reverse engineering of protocols from network traces," in 18th Working Conference on Reverse Engineering, 2011, pp. 169–178.

126. M. Wakchaure, S. Sarwade, I. Siddavatam, and P. Range, "Reconnaissance of Industrial Control System By Deep Packet Inspection," in 2nd IEEE International Conference on Engineering and Technology (ICETECH), 2016, no. 3, pp. 1093–1096.

127. W. Cui, V. Paxson, N. C. Weaver, and R. H. Katz, "Protocol-Independent Adaptive Replay of Application Dialog," in Network and Distributed System Security Symposium (NDSS), 2006, pp. 279–293.

128. J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: Automatic Extraction of Protocol Message Format using Dynamic Binary Analysis," in 14th ACM Conference on Computer and Communications Security (CCS), 2007, pp. 317–329.

129. J. Caballero, P. Poosankam, C. Kreibich, and S. D., "Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering," in 16th ACM Conference on Computer and Communications Security (CCS), 2009, pp. 621–634.

130. J. Newsome, D. Brumley, J. Franklin, and D. Song, "Replayer: automatic protocol replay by binary analysis," in 13th ACM conference on Computer and Communications Security (CCS), 2006, p. 311.

131. W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irun-Briz, "Tupni: Automatic Reverse Engineering of Input Formats," in 15th ACM Conference on Computer and Communications Security (CCS), 2008, pp. 391–402.

132. M. E. DeYoung, "Dynamic protocol reverse engineering: a grammatical inference approach," Air Force Institute of Technology, 2008.

133. A. Trifilo, S. Burschka, and E. Biersack, "Traffic to protocol reverse engineering," in IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009, pp. 1–8.

134. C. Beşiktaş and H. A. Mantar, "Real-Time Traffic Classification Based on Cosine Similarity Using Sub-application Vectors," in Proceedings of the Traffic Monitoring and Analysis 4th International Workshop, 2012, vol. 7189, pp. 89–92.

135. J. Yuan, Z. Li, and R. Yuan, "Information entropy based clustering method for unsupervised internet traffic classification," in IEEE International Conference on Communications (ICC), 2008, pp. 1588–1592.

136. P. Ducange, G. Mannara, F. Marcelloni, R. Pecori, and M. Vecchio, "A novel approach for internet traffic classification based on multi-objective evolutionary fuzzy classifiers," in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2017, pp. 1–6.

137. L. McInnes, J. Healy, and S. Astels, "Benchmarking Performance and Scaling of Python Clustering Algorithms," hdbscan.readthedocs.io, 2016. [Online]. Available: http://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html. [Accessed: 01-Jan-2017].

138. J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation," in Congress on Evolutionary Computation (CEC), 1999, vol. 1, pp. 98–105.

139. J. D. Knowles and D. W. Corne, "M-PAES: a memetic algorithm for multiobjective optimization," in Congress on Evolutionary Computation (CEC), 2000, vol. 1, pp. 325–332.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
11–28–2018	Doctoral Dissertation	Dec 2015 — Dec 2018

4. TITLE AND SUBTITLE

Enabling Auditing and Intrusion Detection for Proprietary Controller Area Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Stone, Brent J., Capt, USA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-DS-18-D-003

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

AFIT/ENG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of this dissertation is to provide automated methods for security researchers to overcome 'security through obscurity' used by manufacturers of proprietary Industrial Control Systems (ICS). 'White hat' security analysts waste significant time reverse engineering these systems' opaque network configurations instead of performing meaningful security auditing tasks. Automating the process of documenting proprietary protocol configurations is intended to improve independent security auditing of ICS networks. The major contributions of this dissertation are a novel approach for unsupervised lexical analysis of binary network data flows and analysis of the time series data extracted as a result. We demonstrate the utility of these methods using Controller Area Network (CAN) data sampled from passenger vehicles.

15. SUBJECT TERMS

cyber-physical systems, cybersecurity, controller area network, reverse engineering, machine learning, artificial intelligence

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Scott Graham, AFIT/ENG
U	U	U	UU	172	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4581; scott.graham@afit.edu