9-2022

Analytic Case Study Using Unsupervised Event Detection in Multivariate Time Series Data

Jeremy M. Wightman

ANALYTIC CASE STUDY USING UNSUPERVISED EVENT DETECTION IN MULTIVARIATE TIME SERIES

THESIS

Jeremy Wightman, Civilian, USAF

AFIT-ENP-MS-22-S-051

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENP-MS-22-S-051

ANALYTIC CASE STUDY USING UNSUPERVISED EVENT DETECTION IN MULTIVARIATE TIME SERIES DATA

THESIS

Presented to the Faculty

Department of Physics

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 Master of Science in Scientific and Technical Intelligence

Jeremy Wightman, BS

Civilian, USAF

August 2022

AFIT-ENP-MS-22-S-051

ANALYTIC CASE STUDY USING UNSUPERVISED EVENT DETECTION IN MULTIVARIATE TIME SERIES DATA

Jeremy Wightman, BS

Civilian, USAF

Committee Membership:

Dr. Gilbert L. Peterson
Chair

Dr. Brett J. Borghetti
Member

Dr. Michael E. Miller
Member

Dr. John S. Knapp
Member

Abstract

Analysis of cyber-physical systems (CPS) has emerged as a critical domain for providing US Air Force and Space Force leadership decision advantage in air, space, and cyberspace. Legacy methods have been outpaced by evolving battlespaces and global peer-level challengers. Automation provides one way to decrease the time that analysis currently takes. This thesis presents an event detection automation system (EDAS) which utilizes deep learning models, distance metrics, and static thresholding to detect events. The EDAS automation is evaluated with case study of CPS domain experts in two parts. Part 1 uses the current methods for CPS analysis with a qualitative pre-survey and tasks participants, in their natural setting to annotate events. Part 2 asks participants to perform annotation with the assistance of EDAS's pre-annotations. Results from Part 1 and Part 2 exhibit low inter-coder agreement for both human-derived and automation-assisted event annotations. Qualitative analysis of survey results showed low trust and confidence in the event detection automation. One correlation or interpretation to the low confidence is that the low inter-coder agreement means that the humans do not share the same idea of what an annotation product should be.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Gilbert "Bert" Peterson, for his guidance and support throughout the course of this thesis effort. The insight and experience were certainly appreciated. I would, also, like to thank LtCol Travis Kornahrens, who was my Squadron Commander through most of my schooling for both the latitude he provided to me in this endeavor and his confidence in my potential. Most of all, I would like to thank my wife, my partner in all things, for her understanding through this long, but rewarding, process.

Jeremy M. Wightman

Table of Contents

Page

List of Figures

Page

List of Tables

Page

ANALYTIC CASE STUDY USING UNSUPERVISED EVENT DETECTION IN TIME SERIES DATA

I. Introduction

In August of 2020, "Accelerate Change or Lose" [1] was a mandate and challenge levied by General Charles Q. Brown Jr., the Chief of Staff of the United States Air Force. Recognizing how the air and space domains are evolving, a battlespace environment characterized by "declining resources, aggressive global competitors, and rapid technology development and diffusion", Gen Brown stated his fears that a failure to change, a failure to adapt, will risk the assured certainty of air, space, and cyberspace dominance enjoyed by the USAF over the last several decades.

Though this document and strategy speaks more to a cultural change within the Air Force, the necessary tenets apply to many Department of Defense (DoD) activities—from acquisition and operations planning to academic research and intelligence. For intelligence purposes, the DoD has long tracked diverse metrics with the hope of gaining important insights into the developing spatial and temporal battlespace.

Analysis of battlespace measures treats them as "signals" that can be abstracted as multivariate time series traces which may exhibit hidden correlated, or causal, relationships. In a more specific domain, the intelligence analysis of cyber-physical systems, especially the processing of unknown signals, could benefit greatly from accelerating analysis techniques. Timely recognition of any deviation from "normal," and recognition of an "event" (either routine or anomalous), could provide critical time-sensitive information necessary to make decisions—possibly with life on the line.

1.1 The Slow Status Quo

Historically, analysis of temporal data signals to classify events has required man-hours, -weeks, or -years of time-intensive, tedious human effort searching for trends then attempting to gain forensic or predictive insight from the data. This analysis also often requires non-trivial domain-specific expertise, compounding both complexity and cost [2] [3]. In modern applications, where data volume and velocity are as concerning as the validity and veracity, the time required for human-in-the-loop processes outpaces needs expressed by the DoD to effectively deliver analysis at scale and speed of relevance. Meeting the veracity challenge will require creative and trusted applications of artificial intelligence and machine-learning in the delivery of new tools, techniques, tactics, and technologies.

One data source that has grown in relevance over the past few years has been network traffic from autonomous cyber-physical systems (CPSs)—systems that integrate digital sensing, computation, control, and networking with physical objects [4]. These can include fixed infrastructure systems such as heating, ventilation, and air-conditioning (HVAC) as well as mobile platforms such as unmanned aerial vehicles (UAVs) or driverless cars.

When looking for events in CPS network traffic, design and test engineers typically know *a priori* how a system's data is collected, stored, formatted, and what parameters/signals are being monitored. Additionally, a system can be extensively tested before deployment—characterizing what is "normal" and what is "anomalous." However, DoD researchers concerned with CPS event/anomaly detection, especially in the scientific and technical intelligence domain, may not have full access to CPS formatting information

due to either proprietary interface control documentation of a manufacturer or data denial efforts by an adversary.

Even after finding the correct data representation, the CPS domain can be exceedingly difficult to analyze. As a system reacts to inputs—either a natural, human, or computer actor—the data changes. There is often no stationarity or trends and any assumptions about the underlying data distribution are often inaccurate. Though CPS data can be correlated across different signal traces, the causal relationship may span several subsystems and separated in time (e.g. engine RPMs lower after brake application). Additionally, DoD analysts are unlikely to have the full testing history of the system available to the original designers. Without significant monetary and time investment, the bulk of any collected data will not be labeled with time coincident "normal" activity (e.g. turns, acceleration, or deceleration) and emergency behaviors or malicious CPS attacks will be exceedingly rare. Unexpected, emergent anomalies may similarly not be observed at all prior to deployment.

When understanding of CPS data is necessary for strategic acquisition and tactical operations, it is critical to find methods to speed up this predominantly human-in-the-loop process to meet the demands of the future battlespace.

1.2 Toward Accelerated, Automated Exploitation

Autonomous CPS analysis is relatively new to academia, with published efforts directed at the automated pre-processing from an arbitrary data stream into formatted time-series data traces. Knapp [2] and Verma, et al. [5] each defined a four-step analysis pipeline

for CPS formatting. Stone [3] demonstrated success with the automated identification of CPS data using unsupervised lexical and semantic analysis. These algorithms provide an initial step toward gaining insight from CPS data. It is the subsequent event detection, classification, and iterative learning as new behaviors are observed, that transforms data from an arbitrary cyber-physical system into human-readable, actionable knowledge.

Though there has been some DoD research into the automated formatting and identification (e.g. [2] and [3]), the majority of event detection research has been performed by civilian researchers in domains as diverse as financial markets [6], medical image anomaly detection [7], and power distribution [8]. Unfortunately, these techniques are predominantly supervised—requiring extensive, and expensive, labeling by domain experts—and typically do not offer the explainability to lead an analyst to root causes without prior knowledge of the interconnectivity between the systems monitored [9]. The DoD does not have the resources, in time or manpower, to rely on these techniques in the rapidly evolving technological battlespace.

Building upon the pre-processing work done by [2], [5], and [3], Figure 1 depicts a combined autonomous pipeline which adds the three additional steps which transforms the data into insight—event detection, classification, and active learning.

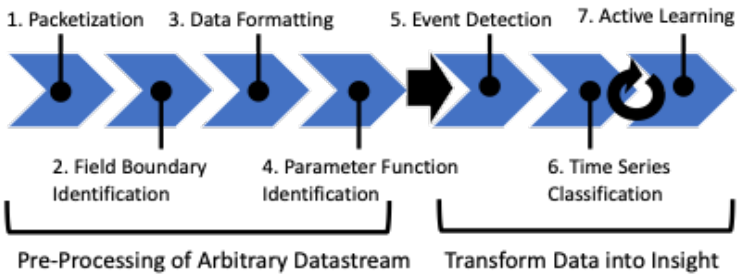


Figure 1: Extended Analytic Pipeline for Arbitrary Datastreams [2].

As a critical part of this exploitation pipeline, the thesis and research presented here focuses on Step 5 – Event Detection. Using expertly pre-processed data collected from a representative dataset, a domain application of unsupervised deep learning techniques is explored for non-parametric, time-correlated, cross-signal event detection in multivariate time series.

1.3 Research Dataset and Domain Expert Pre-processing

Most modern vehicles' on-board electronic operations internally communicate over a Controller Area Network (CAN) bus, tracking the parameters as continuous and discrete pulse-code modulated (PCM), multivariate time series "signals" or "traces." The CAN bus network carries all health/status monitoring and subsystem control traffic between individual components—controlling all vehicle behavior from engine RPM regulation, to power steering, to deployment of safety mechanisms such as air bags or anti-lock brakes. CAN time series data has the non-stationarity, continuously changing behavior that is representative of the target data.

The dataset used for this work was collected using a CAN logger with a personally owned and operated vehicle (POV). The dataset consists of "tailored" driving sessions that only contains particular driving activities (e.g. left turns with stops or engine turn-ons) and 13 "ambient" driving files under clear weather and visibility. Following the process shown in Figure 1, Steps 1 through 4 were performed by applying domain expertise of the researcher, translating the raw CAN data into formatted time series traces, and confirming the identifications through the "tailored" driving set. For initial exploration of the technique

proposed below, a subset of the CAN signal traces was selected as representative of most relevant vehicle activities--namely steering column position, accelerator position, engine RPMs, brake pressure, vehicle speed, transmission state (i.e. park, reverse, neutral, drive, 1, and 2), and turn signal state.

After the data frame was identified, the time series was segmented into overlapping time windows creating a train/test data set of dimensions <number of windows, window length, number of parameters>.

1.4 Hypothesis

For any domain-applicable detection algorithm to find events accurately and reliably, it must autonomously learn what is "normal" and support the detection of activities, events, anomalies, or failures that were not encountered during the initial training phases. Additionally, the method must deliver trusted performance to the human analyst and accelerate the analytic process. However, the cost of labeling the necessary volume of data, and/or collection of sufficiently diverse data to cover all eventualities, are likely prohibitively expensive to use supervised methods.

To address the complex CPS/CAN domain and accelerate decision-supporting analysis, a method is tested which uses a deep learning framework for unsupervised time series reconstruction which uses static error thresholds to autonomously determine both the timing of an event and which data traces led to the detection. In this research, an *event* is as the observed behavior and movement of a physical system at a point in time that deviates from a steady state behavior. Example events for a motor vehicle include a turn, lane

change, engine turn-on, deceleration to stop, or acceleration from stop. Example steady-state behaviors include maintaining a fixed speed, sitting with the engine off, or a long duration stop. For this CPS application, the system undergoes an event which has event indications in one or more of the various channels. As an example, a POV operator will release the accelerator before applying brakes, then the vehicle speed and engine RPMs will lower. Changing activity in the accelerator, brakes, speed, or RPM signals should cross individual reconstruction error thresholds. A change in any of the data traces should be flagged as a part of an event. Steady-state behaviors, such as driving along a straight path, may also involve changes in the accelerator, brakes, speed, and RPM signals as a non-event and should not be flagged.

This research hypothesizes that an unsupervised event detection algorithm consisting of a temporal neural network, error and distance metrics, and a static threshold can deliver demonstrable savings to domain experts in terms of accuracy (F1-score, annotation error, Hausdorff distance) and qualitatively through a case study with domain experts.

The method should apply to any multivariate/multi-sensor time series domain where a user hopes to quickly find events without first teaching an autonomous system what to look for. However, any autonomous system is only useful if it is adopted by the individuals tasked with the domain analysis. A study of potential users, which analyzes their interactions with and trust of the event detection method, is as important as the quantitative performance metrics.

The research questions are:

1. Can unsupervised deep learning methods perform autonomous event detections in multivariate time series data?
2. Does the autonomous system provide sufficient accuracy and reliability to develop enough trust to fit in analysts' existing analysis methods?
3. Are the savings in time, compute, and/or human-analytic burden realized by this method adequate for domain expert adoption?

1.5 Research Methodology

This thesis presents event detection automation system (EADS) and a human usage case study.

This research provides both a quantitative look at autonomous event detection when compared against domain expert annotations and a qualitative survey of the participants' perceptions of delivered analytic utility and time savings. Ten domain experts volunteered to participate in two stages of research focusing first on their legacy event extraction methods in Part 1, and then on capturing their interactions with a selected autonomous event detection method in Part 2. By using several participants, a cross-validation will be accomplished as each overlapping subset of training and test files are used as "truth" data for a different participant.

Following the data pre-processing, Part 1 involved a pre-survey that collected participant demographic information, asked about their perceptions and *a priori* knowledge of artificial intelligence and machine learning methods–in general and particular to their

domain. Next, the domain experts were presented with a subset of the "ambient" driving data and tasked with performing their manual event extraction process. The study established a baseline for the time required for their legacy methods and the inter-coder agreement for the events.

Results from Part 1 were used to select the unsupervised event detection model to be used during automation. Generally, the system took the windowed, strided data and trained a deep reconstruction model which recreated the input data after first passing it through a lower dimensional space (called the latent space). Next, the reconstruction was subtracted from the original trace to create windowed time series reconstruction error traces and distance metrics were calculated between the latent space representations at each timestep. Through either a mean-based or histogram-based method (based on the data distribution) calculated from the training data, a simple static threshold was used to generate candidate events in the reconstruction error and distance space. Convolutional neural net (CNN) autoencoders [10], recurrent neural net (RNN) with long, short-term memory (LSTM) autoencoders [11], and self-attention transformers [12] are evaluated. Using a researcher-generated baseline file, the overall system was tested with a coarse grid-search through the data window size; model architectures; and basic threshold techniques to optimize the event detection algorithm on the baseline dataset. Based on model performance, the researcher selected a set of model and event detection hyperparameters to be subsequently presented to the participants.

In Part 2, the participants were asked to validate the events output by the autonomous systems. For each participant, their Part 1 files were used as training examples for a user-specified reconstruction model, then the other files are used as their test data.

These files were be passed through the autonomous system described above and the unsupervised algorithm will present candidate events to the participants. Next, the participants were tasked with correcting the events autonomously generated by the algorithm. Lastly, they were asked to fill out a survey about their perceived trust, confidence, speed of use, and analytic utility of the algorithm.

Ultimately, the quantitative data from the model outputs compared against expert annotations and the qualitative responses to surveys were used to evaluate the autonomous event detection method as it pertains to the analysts' application domain.

Human research approvals and IRB information can be found in Appendix A.

1.6 Outline of Following Chapters

The remaining chapters show current and proposed applications for unsupervised, non-parametric event detection in multivariate time series. Chapter 2 describes Related Works, briefly discussing legacy time series decomposition and autoregressive techniques, before moving on to statistical changepoint detection algorithms, and then addressing their deficiencies when applied to the CPS/CAN domain (e.g. non-stationarity, assumption of underlying data distributions, and hyperparameter search spaces). Next, the chapter considers several deep learning techniques and their application to the CPS/CAN domain, then concludes with a discussion of qualitative case study methods. Chapter 3 further discusses the Event Detection Automation System and Chapter 4 discusses the Case Study methodology. Chapter 5 discusses the Results of Part 1 (both the pre-survey and initial annotation task), coarse model search for the autonomous event detection (across several

architecture choices), then presents the results of Part 2 (participant validation of model outputs and post-survey). Lastly, this research concludes with Chapter 6 and discusses Conclusions and suggestions for Future Work.

II. Related Work

In the age of Big Data and the Internet-of-Things (IoT), there is simply too much data for humans alone to analyze. It is critical to develop methods to quickly understand when a monitored system is "normal" and to accurately, reliably, and responsively detect changes–or events–so decision-makers can react appropriately.

Event detection methods are typically specific to an application domain, with special consideration for data type, characteristics, and target detections (i.e. what types of activities an analyst is hoping to find). These event detection methods are also intimately connected with forecasting and prediction. If an analyst can presciently understand what will happen, one could anticipate change and shape events. Without that clairvoyance, researchers can only make predictions, either of the underlying data generative process or a future measurement and observe events which deviate from that prediction.

This thesis presents event detection in time series data (deviations from steady-state in data observations ordered in time). The specific domain being analyzed in this research focuses on timestamped measurements from vehicle instrumentation. These outputs are called *signals*. When collected (or sampled) over time, these continuous and discrete data points (or samples) are called data *traces*. Multiple traces can span different *channels*, where each channel corresponds to the output of a single sensor. Signals, traces, and channels are all observations of the underlying system measurements ordered in time. The terms are often used interchangeably in the target domain and will be used similarly in this thesis.

This chapter begins by defining key terms such as "event" and "time series data" before discussing the formal event detection problem. Next, methods for time series

modeling lead into specific event detection methods. Quantitative evaluation metrics are also discussed. This Chapter concludes with discussions on qualitative survey methods, dealing with domain expert generated annotations, and considerations for how detection methods could fit into the analysis workflow shown in Figure 1.

2.1 Event Detection Terminology

The event detection literature has no single agreed upon definitions and use terms like anomaly, outlier, rare occurrences, and novelty. Some use the terms interchangeably, and others attribute small nuances which allow their distinction. According to [13], these definitions do share common themes:

1. The distribution of these observations are significantly outside the general distribution of the data.
2. These points represent a small portion of the overall dataset, i.e. the majority of the data is considered "normal."

Braei and Wagner [13] collects most of these terms and calls them anomalies (leaving some distinction for novelty). However, labeling a change in behavior as an anomaly may be limiting–there are connotations of strictly unexpected/unintended activity. Steady-state activity is defined as non-event activity. In the case of a POV, this could be a long duration stop or regular straight driving. There are a host of applications where any deviation from a steady-state is of interest.

This thesis focuses on data that has a temporal component, where measurements or observations are ordered in the time dimension and uses a broad definition of event that

includes all anomalies, faults, novelty, etc. but also normal change activity. For example, a thermostat failure would be a fault, but an operator input that changes the temperature is a normal occurrence. Both result in a temperature alteration, one intentional the other unintentional, but in this work, both are considered events.

Additionally, the broader definition of event may affect the distribution of normal and anomalous labeling of the data. When including regular activity, a signal may be changing from one normal activity to another and may spend significant time in event states. Algorithm selection needs to accommodate this different distribution for applicability to the CPS domain.

2.1.1 Types of Events

This research limits the environment of defining an event to temporal digital data in a structured format consisting of attributes and labels where there is a time associated with a sample of labels assigned to the attributes.

Braei and Wagner [13] describes three common types of anomalies. Using their definitions and applying them to this domain, events exist as:

1. **Point events**: a single data point sample that deviates significantly from the rest of the data. This could be a single large transaction which exceeds normal spending habits.
2. **Collective events**: a sequence of points which are labeled as part of an event when a single measurement may or not be classified this way. This could be a month of higher than normal temperatures.
3. **Contextual events:** a sequence of points which are considered an event under one context but not another. A simple example could be snowfall near the equator. Snow is not uncommon near the poles, but rare that far south.

The term event already includes a time-orientated connotation and similarly suggests most detections will likely be collective or contextual. The expected types of events in a dataset or an analyst's target detection (what kind of event the analysis is looking for) influences the type of algorithm to be used. Some methods are only able to detect point events and miss collective/contextual events. Depending on the domain application, changing time scales or aggregating samples (reducing sampling rate) may change the available options. Decisions about data preprocessing and appropriate event detection methods are made while considering the type of events to be detected.

2.1.2 Types of Detection Methods

As the size and utility of time series data increases, the amount of research has increased accordingly. Braei and Wagner [13], Mahmoud and Mohammed [14], Goldstein and Uchida [15], Aminikhanghahi and Cook [16], and Truong, et al. [17] have all published event detection methodology surveys, each with slightly different foci [14] [15] [16] [17].

Like defining an event, it is important to establish a vocabulary to describe event detection methods. This research focuses on temporally ordered data, and the definitions focus on terms applicable to time domain analysis.

2.1.2.1 Detection vs. Segmentation

Analyzing the behavior of a temporal sequence for events can fall into two categories: detection and segmentation [17]. Event detection is looking for the characteristics of a transition, a relatively short duration period of behavior or class

instability. This could be a transmission gear shift when a vehicle is accelerating, an injected transition from one state to essentially the same state, or could be a change from drive to park- completely different vehicle activities. Segmentation examines the behavior of the data and identifies changes in that behavior. Music segmentation is one example where the event occurs between musical movements, e.g. between verse and chorus.

An event detection could be characterized as a short transition segment before the data returns to another known steady-state. However, this researcher believes the primary contrast is in the follow-on classification, the subsequent step following detection in Figure 1. A segmentation task would classify the activities between detections and the detection task would classify the transitions. The most robust systems may perform both tasks.

Some statistical methods, like changepoint detection, bridge this distinction by looking for a singular data point (detection) that separates two periods of activity (segmentation). However, many of these methods are incapable of finding collective or contextual events.

This thesis focuses on the detection of collective and contextual events at the transition periods between steady-state activities.

2.1.2.2 Statistical vs. Machine Learning Methods

There are two main categories of event detection methods for time series [13] [18]: statistical approaches which assume the data is generated by some specific underlying model and machine learning approaches which attempt to learn a representation directly from the data. The method requires a researcher to make assumptions about the generative process before analysis. Machine learning assumes the underlying model is not required if

the method is able to learn an accurate, representative data model. Braei and Wagner [13] further segments machine learning into classical methods and artificial neural networks (or deep learning).

Statistical and machine learning approaches are also related to parametric and non-parametric methods. Aminikhanghahi and Cook [16] discuss their differences. Parametric algorithms specify a functional form of the solution and then estimate the unknown parameters of the model based on the data. These methods allow the training data to be discarded once the parametric model is created. Conversely, nonparametric approaches do not make assumptions about the underlying function, must retain the training data through inference, but have been shown to scale better with larger datasets.

Most machine learning techniques are parametric where a model is specified with a certain number of trainable parameters. Then, those parameters are learned by iteratively testing a model, calculating an evaluation metric, then attempting to adjust the parameters to better fit the evaluation criteria. Some statistical models, e.g. change point detection algorithms, can be parametric or nonparametric.

2.1.2.3 Online vs. Offline Detection

Based on the intended domain application, there are two ways of dealing with data [16] [17]. The first, *online*, aims to analyze data in near-real time as it is collected, sometimes called *streaming data*. Essentially, online methods run concurrently with the processes they are observing, analyze each data measurement as they are made, and attempt to detect events as soon as they occur. Conversely, *offline* methods wait until an entire dataset is collected and then retrospectively analyze the data for change events.

Truong, Oudre, and Vayatis [17] mention that online methods are often referred to as event or anomaly detection and offline methods are called signal segmentation. This thesis uses the definition where detection is attempting to find a transition point and segmentation focuses on the behavior between detections, both of which can be done online or offline.

Online and offline methods can also reference the types of training available to a learning system [19, pp. 14-17]. *Batch learning* is done with all currently available data, inherently an offline method. To add understanding about new data, the dataset grows and the model retrained. *Online learning* techniques can be trained incrementally by feeding data to the system as it is collected. Online methods benefit from:

- Lower data storage needs because it can discard incorporated data.
- Lower training time for new data because it only needs to learn from the additional measurements, observations, or samples.
- Better performance on large datasets because the incremental training does not require the entire dataset to be held in main memory for computation.

However, online learning requires more monitoring after application launch because bad data ingestion may cause gradual decline in performance [19, p. 17].

This research focuses on offline methods to better align with the CPS data target domain of the participants.

2.2 Time Series Data

Time series data is a set of ordered measurements, labels, observations, etc. sequentially recorded over time [13] [14] [15] [16] [17]. This time domain can be continuous. However, most current applications utilize digital computing and require

discrete time samples. In both cases, the rate of observations, the sample rate, can be at constant, irregular, or variable. This is further complicated when trying to draw insight from multiple, disparate, and diverse sources (e.g. different sensors)–each with their own dynamic ranges, sampling rates, and possibly data formats.

2.2.1 Time Series Definition

By definition [13], time series is a realization of a stochastic process $Z(\omega, t)$ where ω belongs to a sample space and t belongs to an index set. Z is a random variable in the sample space if t is fixed. Therefore, a time series is defined as:

$$T = (t_0^c, t_1^c, \dots, t_t^c), c \in \mathbb{N}_+, t \in \mathbb{N} \quad 1$$

where T is the time series, and C is the number of channels or dimensions of the dataset. This represents the number of disparate time series measurements collected from different sensors, phenomena, observers, etc.

Unlike other datasets, time series observations depend on both the cross-dimension relationships and the order of the time sample set [13]. Statistically, time series signals are not independent and identically distributed. After a given instance in time, t_n, the range of likely values at t_{n+1} are dependent upon the previous instances within the set (except in the presence of noise).

In this paper, only discrete time series are considered. The domain is further restricted to a constant, equal time sampling–ensured during data preprocessing, normalization, and interpolation.

2.2.2 Time Series Characteristics

Event detection in time series data assumes that historically collected samples can representatively convey the underlying process and carries useful information for the description of phenomena. Essentially, this means an observer (human or computer) can analyze the past to make educated inferences about what did or will happen.

There are several characteristics which describe a time series: scale, trend, seasonality, cycle, residuals, stationarity, and noise. Understanding these properties is critical because the data behavior impacts the accuracy and effectiveness of various detection algorithms.

2.2.2.1 Time Scale

The first consideration when analyzing time series data is the time scale to be observed. Typically, this is defined by the sampling rate of the data and the expected duration of an anomaly or event. For example, one sample a decade is not helpful when observing decaying cesium atoms in an atomic clock. Alternatively, sampling every cycle of a computer processor is likely unnecessary, and wasteful with data storage, when looking at monthly sales numbers at a sporting goods store. The scale also influences which of the time series characteristics is observable and/or how they are presented.

2.2.2.2 Time Series Components

From the originally collected time series data, the underlying signal can be modeled as a multiplicative (Equation 2) or additive (Equation 3) composition of 4 components: trend, season, cycle, and everything else (residuals).

$$Y_t = T_t \times S_t \times C_t \times R_t \quad (2)$$

$$Y_t = T_t + S_t + C_t + R_t \quad (3)$$

where Y_t is the original data, T_t is the trend value, S_t is the seasonal value, C_t is the cyclical fluctuations, R_t is the residual data all at time sample t.

Trend

A trend is a non-constant mean in a time series. A trend can increase or decrease over time described as a positive or negative trend, respectively.

Seasonality

A time series demonstrates seasonality when there is a behavior with a fixed periodicity, typically less than a year. Seasonality is named after seasonal factors such as decreased temperatures during winter months or longer days during summer. These recur at a regular, expected, and predictable timeframe–hence the fixed periodicity.

Cycle

Cyclical variation in a time series occurs from periodic phenomena that have a variable period or whose changes extend beyond the observed time window/scale.

Residual

Once the trend, seasonality, and cycle are removed, what remains is typically the data of interest. The residual signal represents the variation resulting from unexpected or irregular changes in the underlying data. Using customer sales as an example, this residual could be due to additional environmental factors (e.g. rain), normal small variation over

time (e.g. a few more customers on a given day), or represent the actual event that is the intended focus of an analysis (e.g. a surge in sales due to a new product launch).

2.2.2.3 Stationarity

When a time series is the same over every time interval, i.e. the distribution at any sample is equal, then the data is considered stationary [13]. A stationary time series has: a constant mean (no trend); a constant variance; no seasonality (no periodic variation); and a constant autocorrelation over time (the expected value is constant when a signal is compared against a time lagged version of itself).

2.2.2.4 Noise

When any sensor makes observations or real measurement, there is a potential for error. These errors can be from unintended interference, natural phenomena, or man-made processes. Regardless, this error, called noise, is an unintended injection which influences the time series collection–and could influence the perceived/calculated underlying process.

Like anomalies/events, noise can be a point, collection, or contextual. Point noise is observed over a single sample, possibly a bit error from signal transmission. Collective noise is longer duration and covers multiple time points. The same phenomenon can also be characterized as a point or collection depending on the time series scale. For example, an increase in sales over a week will appear as a collection when sampled daily but a point anomaly when sampled monthly. The last type of noise, contextual, is dependent on the target detection. An example here could be mislabeled data from human annotations. If a researcher is analyzing the rate of erroneous annotations, then these data points are the

signal of interest. If the project hopes to use correct labels to draw inference, then those same points are noise (called *class noise*).

White noise is a special case which is uncorrelated over time from a fixed distribution with a constant, zero mean and a finite variance [13]. White noise is stationary and there is no dependence between two observations.

2.2.2.5 Labeled or Unlabeled

A labeled dataset has an associated annotation for each element of the dataset [13]. For pictures of animals, a label may take the form of $\{cat, not\ cat\}$. In the case of event detection in time series, this label is typically a binary $\{normal, anomaly\}$ but may have multiple possible labels associated with different event types $\{event\_type1, event\_type2, \dots, event\_typeN\}$. The presence of labels directly impacts the possible choices of machine-learning algorithms. If the dataset is labeled, supervised methods typically offer the highest accuracy. Partially labeled sets can utilize semi-supervised methods that can allow the propagation of labels to similar samples. Without labels, an analyst must use unsupervised methods.

2.2.2.6 Univariate Versus Multivariate

A univariate trace is when a single time series is considered for analysis. Common examples include stock prices, sales dollars, or environmental temperature. Multivariate datasets include multiple time series dimensions such as different sensors, observers, or other sources. Multivariate data can be measuring the same phenomena from different locations (e.g. seismographs), different aspects of the phenomena (e.g. color wavelengths),

or different but contextually related phenomena (e.g. stock index prices and inflation). Regardless, analyzing a multivariate dataset is usually done to determine the relation between the channels as they influence an activity of interest–an event.

2.2.2.7 Cross-dimensional Correlation or Causation

The final consideration when analyzing time series data specifically relates to multivariate data. Namely, how do the channels behave together or influence each other across dimensions? Are changes correlated or does one channel cause variations in another, i.e. a causal relationship? This relationship may require *a priori* domain knowledge or assumptions about the underlying processes. Sometimes, determining this correlation is the entire purpose of an analysis (e.g. predicting the price of pork belly commodities based on the orange juice market). Other times, the relationships help to explain the causality of an anomaly or event.

2.3 Framing an Event Detection Problem

Event detection solutions are unique to an application domain. Different methods perform better on data with certain behaviors, characteristics, or properties. Some algorithms may be incompatible entirely and unable to give any results. Therefore, a strong foundational knowledge of the application domain is necessary to shape event detection research. This initial problem framing could save enormous amounts of time and comes in two parts: knowing your goal and knowing your data.

2.3.1 Event Detection Problem Definition

As the classic stoic philosopher Seneca said, “If a man knows not which port he sails, no wind is favorable.” [20] Therefore, defining what a time series analysis project hopes to accomplish is a necessary first step.

The first step is to determine the types of events to be found and an associated detection target. Essentially, what is the purpose of the event detection process and what indicators may appear in the data? Braei and Wagner [13] discusses two types of detection targets:

1. Detection based on prediction of the time series.
2. Detection based on unusual shapes in the time series.

Prediction and shape detections appear in both statistical and machine learning methods. Knowing the intended target for the application domain will drive technique selection.

2.3.2 Data Domain Characteristics

If an analyst is unfamiliar with a dataset, it is critical to perform an initial domain exploration to understand:

- Is the data univariate or multivariate?
- Are labels present for some or all the data?
- What types of events can be expressed by this data?
- Are there trend, cycle, or seasonal patterns?
- How clean or noisy is the data?
- How much data is accessible?

All of these questions will help answer the most important question: Is this data likely suitable to accomplish an event detection problem? Once answered in the affirmative, the questions can additionally drive data preprocessing, time series modeling, and event detection algorithm decisions.

2.4 Time Series Modeling

Though not necessary for all applications, time series forecasting and prediction, can be intimately connected with event detection. In cases which use time series modeling, a representative model must be assumed or learned so that a system can know what data is expected and whether new data matches that expectation, regardless of the prediction- or pattern-based target detection selected.

Like the event detection algorithms themselves, there are two methods for time series modeling: statistical and machine learning.

2.4.1 Statistical Models

Statistical models start from an assumed data distribution and underlying generative process. If the assumption is accurate, then predictions can be made about past and future data based on that model. Unfortunately, statistical models are either univariate or computationally costly for multivariate data. Most of these statistical techniques also require signal stationarity so it is important to know the data characteristics and transform the data if necessary.

2.4.1.1 Classical Decomposition

When a time series exhibits a strong cycle, trend, and/or season, each of these components can be modeled and used for a baseline data prediction. Using either the additive or multiplicative models in Equation 2 and Equation 3, developing the time series model requires the calculation and removal of each component individually then adding them back during prediction stages.

Moving average (MA) calculations are often used to smooth out the short-term fluctuations and identify long-term trends–an estimation of the cycle and trend [14].

$$\hat{T}_t = \frac{1}{m}\sum_{j=-k}^{k} xt + j \quad (4)$$

Where $\hat{T}_t$ is the trend and cycle, $m = 2k + 1$ and varies over a window size of $2k$ sequential measurements, and x is the value of at time t.

Simple exponential smoothing (SES) is another method for determining the cycle-trend [14].

$$\hat{m}_t = \alpha X_t + (1 - \alpha)\,\hat{m}_{t-1}, s = 2 \dots n \quad (5)$$

for a fixed $\alpha \in [0,1]$ and $\hat{m}_1 = X_1$.

Seasonality can be removed by averaging the values at a given seasonal timestamp (e.g. every Tuesday) or subtracting the value at a given time point by the previous (called differencing) [19, p. 506]. There may be multiple seasons to be discovered and removed.

$$X'_t = X_t - X_{t-n} \quad (6)$$

where n is the length of the season.

Though these techniques can be used for predictions by themselves, they may also be used as pre-processing for other time series modeling which may require data in a particular form.

2.4.1.2 Autoregressive with Moving Average Models

There are several types of time series models which have autoregression as a foundation.

Simple Autoregressive Model (AR)

An autoregressive model is a simple linear model where the current value of a time series is based on a finite set of previous values.

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t \quad (7)$$

where X_t is the current value of the stochastic process, coefficients $a_1, \dots, a_p$ and c can be approximated from the data. An autoregressive process of order p, $AR(p)$, has a preceding window length equal to p. The error ϵ_t is considered uncorrelated and used for event detection.

Moving Average Model (MA)

The moving average is a linear transformation of the last q observations of a time series and the last q errors $\{\epsilon_t, \epsilon_{t-1}, \dots, \epsilon_{t-q}\}$, also called a q order moving average or $MA(q)$.

$$X_t = \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \mu + \epsilon_t \quad (8)$$

where μ is the mean of the time series and coefficients $\{b_0, b_1, \dots, b_q\}$ are learned from the data.

Autoregressive Moving Average Model (ARMA)

A more sophisticated extension combines both the AR and MA.

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \epsilon_t \quad 9$$

In this algorithm, the model is dependent on the last p observations and q errors. These control whether the model over- or under-fits the data and are learnable parameters using correlograms [13], leave-one-out cross-validation [21], and the Box-Jenkins Method [22].

Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a generalization of the ARMA model which removes the stationarity precondition.

$$X_i' = X_i - \sum_{j=1}^{d} X_{i-j}, \forall i \in \{1, \dots, T\} \quad 10$$

Effectively, this method adds a d parameter which defines the number of times an ARMA model is differenced to remove cycle-trend or season ($d > 1$ if the cycle-trend is non-linear).

2.4.1.3 Other Statistical Models

The autoregression-based models are only a few of the possible statistical methods. Some methods expand on the simple exponential smoothing to double and triple exponential smoothing (DES, TES) [23]. Time series outlier detection using prediction

confidence interval (PCI) [24] forecasts the next time stamp measurement by using a non-linear weighted sequence of previous data.

There are others but the above sample of statistical models represent examples that are well-supported by historical usage and research.

2.4.2 Machine Learning Models

Where statistical models pre-assume an underlying statistical process, machine learning methods assume that the generative process is irrelevant if a representative data model can create accurate predictions. This data model is learned directly from the data used to train the model.

Though techniques exist which perform well for event detections, classical machine learning methods such as linear/logistic regression, clustering, or classical dimensionality reduction are known to be ill-suited for time series modeling and the associated forecasting/prediction tasks. This is because each timestamp in a fixed window is treated as an individual feature, with multivariate data appearing as concatenated feature sets of the same window length. With the large number of features, these classical methods suffer from the Curse of Dimensionality which shows a reduction in performance due to increasingly sparse feature spaces as the number of dimensions grows [19, p. 214].

However, machine learning techniques using artificial neural networks, or deep learning, have been shown to be very effective for modeling, predicting, and forecasting time series data.

2.4.2.1 Artificial Neural Networks

Neural network-based machine learning has garnered enormous popularity and success in recent years. This increased prevalence has been observed in domains as diverse as stock prediction, natural language processing, and image recognition. Though most of the early success with artificial neural networks (ANNs) was seen in computer vision tasks such as object detection and classification, there is a growing body of research applying these machine learning techniques to time series applications such as forecasting, segmentation, and classification [13] [14] [15] [16] [17] [25].

Mahmoud and Mohammed [14] lists a few advantages of all ANNs that include:

1. Able to learn very complex, non-linear mappings from the input to output to include time series
2. Able to handle multiple types and dimensions of input data plus multivariate time series
3. Do not require stationarity in time series inputs

However, this is at the cost of more powerful computing hardware, time cost of possible long training times, a need for lots of training data to be generalizable, and dependent on weight initialization and gradient descent methods–possibly suffering from local minima in the training optimization contour and not finding a global minimum [19].

The basic unit of an ANN, the *perceptron*, is based on an older artificial neuron called the *threshold logic unit* (TLU), shown in Figure 2.

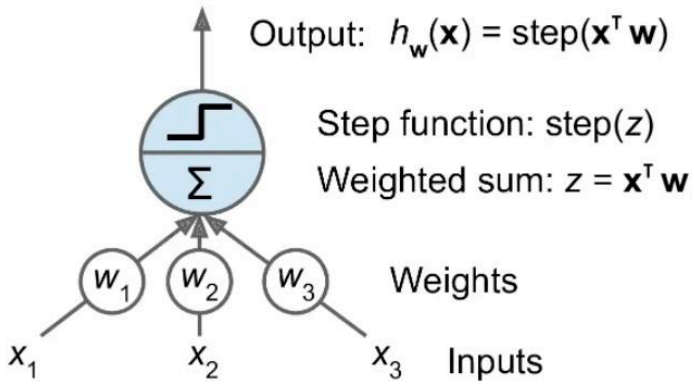


Figure 2: Threshold Logic Unit (TLU) [19, p. 284].

Essentially, the TLU multiplies the input matrix by a matrix of weights. This output is then passed through a non-linear step function, more generally called an *activation function*, to calculate the TLU output. The final calculation may or may not be close to the desired output. To determine that difference, a loss function (also called a cost function or performance measure) is applied to the input and output. Common loss functions include *mean square error* (MSE) [19, p. 121], shown as Equation 11; *root mean square error* (RMSE) which gives higher weight to larger errors [19, p. 39], show as Equation 12; and *mean absolute error* (MAE) which allows for more outliers [19, p. 41], shown as Equation 13.

$$MSE(\boldsymbol{X}, h) = \frac{1}{n}\sum_{i=1}^{n}\left(h(\boldsymbol{x}^{(i)}) - y^{(i)}\right)^2 \quad 11$$

$$RMSE(\boldsymbol{X}, h) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h(\boldsymbol{x}^{(i)}) - y^{(i)})^2} \quad 12$$

$$MAE(\boldsymbol{X}, h) = \frac{1}{n}\sum_{i=1}^{n}\left|\left(h(\boldsymbol{x}^{(i)}) - y^{(i)}\right)\right| \quad 13$$

where n is the number of data points, $\boldsymbol{X}$ is the total input matrix, $\boldsymbol{x}^{(i)}$ is the i^{th} feature vector of the input, $y^{(i)}$ is the target value, and $h(x)$ is a function (typically the activation function discussed later).

If these losses are large, the TLU will require adjustments to the weights matrix to optimize the resulting output compared to the target. This is called learning.

Like classical machine learning methods, ANNs iteratively learn a data model directly from the training samples–the underlying process does not matter if the model can accurately make predictions. The most common learning method, for both classical and artificial learning, is called gradient descent which takes the partial derivative of the loss function (with respect to the weights), then uses that resultant vector to update the weights matrix for the next learning iteration, called an *epoch*.

$$\boldsymbol{\theta}^{(next\ epoch)} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{14}$$

where $\boldsymbol{\theta}$ is a matrix of features, $L(\boldsymbol{\theta})$ is a loss function (e.g. MSE), and η is a learning rate which controls the speed of descent.

Further extensions on this basic method add a level of randomness to the weights update (stochastic gradient descent [19, p. 124]), train on portions of the data at a time (batch gradient descent [19, p. 121]), and momentum [19, p. 351]. The most sophisticated optimizers combine several of these extensions to include the Adam optimizer [19, p. 356], the adaptive momentum estimator, which combines the stochastic gradient descent, momentum function, and *root mean square propagation* (RMSProp).

$$\boldsymbol{\theta}^{(next\ epoch)} = \boldsymbol{\theta} - \widehat{\boldsymbol{m}}_{\boldsymbol{t}} \left(\frac{\eta}{\sqrt{\hat{\boldsymbol{s}}_{\boldsymbol{t}}} + \epsilon} \right) \ where\ \widehat{\boldsymbol{m}}_{\boldsymbol{t}} = \frac{\boldsymbol{m}}{(1-\beta_1^t)}\ and\ \hat{\boldsymbol{s}}_{\boldsymbol{t}} = \frac{\boldsymbol{s}}{1-\beta_2^t} \tag{15}$$

where $\boldsymbol{m} = \beta_1 \boldsymbol{m} - (1 - \beta_1)\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ and $\boldsymbol{s} = \beta_2 \boldsymbol{s} + (1 - \beta_2)\nabla_{\boldsymbol{\theta}}^2 L(\boldsymbol{\theta})$ with β_1 and β_2 are the momentum decay and scaling decay, respectively.

The final consideration is how the weights matrix is initialized. Random initialization tends to work well. However, others such as Glorot, He, and LeCun may be recommended depending on the type of activation function used [19, pp. 333-334].

2.4.2.1.1 Multi-Layer Perceptron (MLP)

A single TLU with a simple step-wise function has very limited utility. It is when many TLUs, with more sophisticated activation functions, are chained together that give ANNs their flexibility and power. These architectures are called multi-layer perceptron (MLP), a simple example is shown in Figure 3.

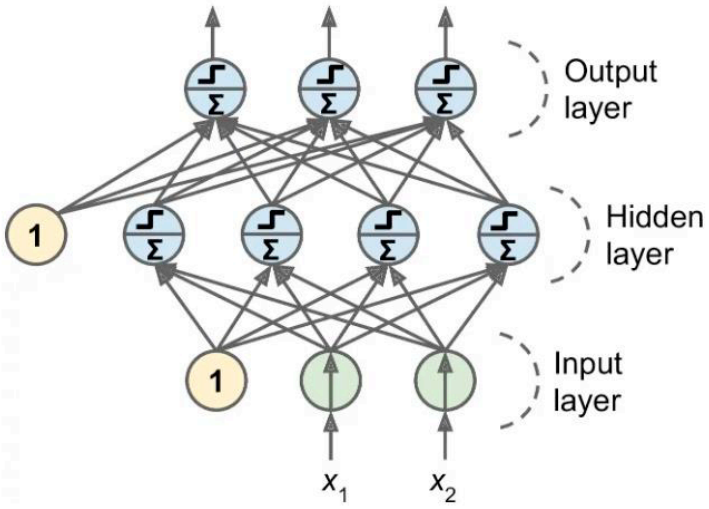


Figure 3: Multi-Layer Perceptron [19, p. 289].

As shown in the Figure above, all MLPs have an input and output layer typically with one or many hidden layers between (the bias term is always output as 1). Each layer has a certain number of neurons with a shape corresponding to a layers' function and the

general purpose of the network. The input layer always has a shape identical to the input data. The output layer is shaped by the desired output of the ANN. The hidden layers are those between the input and output layers. If every neuron in a particular layer is connected to every neuron in another, they are called a *dense* or *fully-connected* layer.

The size of each layer with its prescribed number of neurons, sometimes called the width, and the number of hidden layers, called the depth, are critical hyperparameters to test when optimizing an ANN-based machine learning application. The increased depth of the MLP is why this framework is called a *deep neural net* (DNN) and the learning method is called *deep learning*.

In addition to hidden layers, MLPs and other DNNs have more options for the non-linear activation functions $g(z)$. These include:

- Sigmoid (aka Logistic)

$$g(z) = \frac{1}{1+e^{-z}} \quad 16$$

- Softmax

$$g(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 17$$

- Hyperbolic Tangent (tanh)

$$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \quad 18$$

- Rectified linear unit (ReLU)

$$g(z) = \max(0, x) \quad 19$$

where $\boldsymbol{z} = \boldsymbol{x}^T\boldsymbol{w}$, $\boldsymbol{x}$ is the input matrix and $\boldsymbol{w}$ are the matrix of weights.

The greatest difference between the simplest TLU and a true neural network is how the hidden layer weights are updated during learning [26]. This revolutionized the utility of DNNs by developing a method to efficiently determine how much of the final loss calculation at the output layer is attributable to each hidden layers' weights. This is called

backpropagation. Essentially, the weight update at a given layer is a function of the remaining loss from the previous layer (working backward from the output through the hidden layers).

$$\frac{\partial L(\theta)}{\partial w_{ij}^{k}} = g'\left(a_j^k\right) o_i^{k-1} \sum_{l=1}^{r^{k+1}} w_{jl}^{k+1} \delta_l^{k+1} \qquad 20$$

where the i, j are matrix coordinates for layer k and r^{k+1} are the number of nodes in that layer. Essentially, the error δ_j^k at layer k depends on the errors at the next layer $k+1$.

It is backpropagation which makes ANNs computationally efficient enough to find application in such varied domains.

There are several additional variations on the ReLU activation such as the leaky ReLU, exponential linear unit (ELU), and Scaled ELU (SELU) [19, pp. 335-338].

Each activation function has properties which may be desirable for certain machine learning applications. A sigmoid activation ensures the output varies between [0,1]. It is often used for binary classification but could also be used as a probability. A softmax in the output layer allows the system to apply multiple class labels to the data. The tanh activation has outputs which vary between [-1,1], and the ReLU and its variations do not saturate at extremely high or low gradients (called the *exploding gradient* and *vanishing gradient* problems).

Lastly, the output layer is directly influenced by the intended output of the DNN. The choice of the final activation function has an influence on the dynamic range possibilities of the output. In a time series context, the number of neurons in the output could correspond to the number of timestamps the system is intended to predict or classify.

Though the hidden layers affect the performance of the system, it is the output layer which is more intimately tied with the functional application.

2.4.2.1.2 Convolutional Neural Networks (CNN)

MLPs, though capable of learning very complex mappings between input and output, still require a 1-dimensional input series to match the long line of single neurons in each hidden layer [19, p. 447]. For multivariate data sets, any inputs must be flattened to a vector. This can cause the MLP to lose spatial relationships between input channels or dimensions (e.g. multiple color levels for the same pixel in an image).

Convolutional neural networks (CNNs) are another extension to basic ANNs which were developed to act similarly to the visual cortex in mammals. Biological neurons look for small patterns in the visual field then aggregate them into higher-level features and object recognition (classification). Artificial CNNs act similarly by using stacked *convolutional* and *pooling* layers to reduce the input into lower-level feature maps. As the data passes through deeper hidden layers, the receptive field of each neuron increases allowing it to recognize more complicated patterns. CNNs also allows $d > 1$ dimensions (i.e. multivariate) for the input layer and can learn the spatial relationships missing from a simple MLP [19, pp. 446-449]. Figure 4 shows an example of a multi-channel input with multiple feature maps per hidden layer.

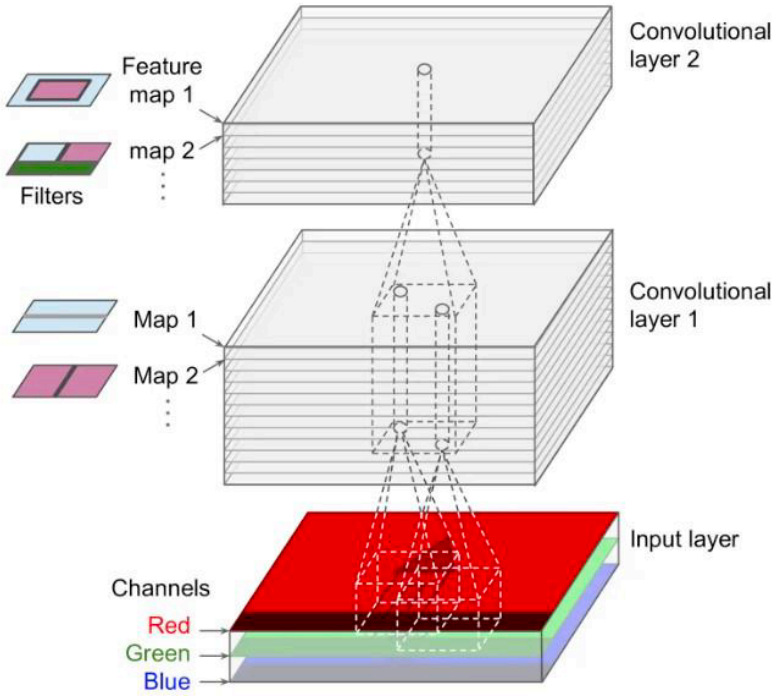


Figure 4: Multi-Channel Convolutional Neural Network (CNN) [19, p. 452].

The key innovation of the CNN architecture is the convolutional layer. Basic MLPs are fully-connected which creates a large number of parameters and makes them computationally expensive for deep networks, requiring additional training time. Convolutional layers are partially-connected and use a sliding window of a one- or multi-dimensional *filter*, also called a *kernel*, to pick out the features in the previous layer which activate the filter the most. For example, a vertical kernel will find all the vertical features. More complicated patterns can be recognized by using many different filters of different patterns. The kernel window slides along all dimensions of the previous data tensor. Padding the input data can ensure that information is not lost at the edges of the sample. *Valid padding* implies there is no padding, the kernel starts at the edge of an input, and the

resulting output will be a smaller dimension. *Same* and *zero padding* add p extra numbers to the edges of the input such that the resulting output is the same shape as the input.

The partially-connected nature of CNNs reduces the overall number of trainable parameters which typically corresponds to a decrease in overall training time. The number of parameters can be further decreased by skipping some kernel windows with a given *stride*, essentially mapping a larger convolutional layer to a smaller layer. Figure 5 shows this reduction in dimensionality.

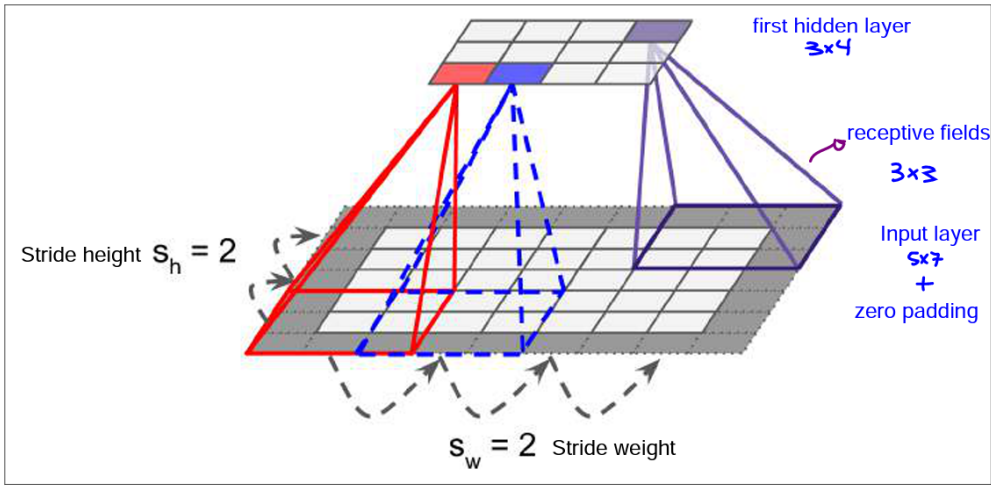


Figure 5: Reducing Dimensionality with Stride of 2.

Another common feature in CNNs are pooling layers. These layers subsample the input to reduce the number of parameters even more, therefore lowering the computational load and required memory as well. This also reduces the likelihood of overfitting. *Max*, *min*, and *average* pooling [19, pp. 456-457] are used though max pooling appears the most common [14].

The last layer before the output in many CNN architectures are dense, fully-connected layers. These layers at the output finish the CNN mapping to either a set of numerical outputs or classes.

Historically, CNNs have been known for computer vision tasks [13]. However, recent applications have seen success with time series data [11] [25] [27] [28] [29] [30]. While 2D or 3D kernels are common for image classification, 1D convolutional kernels and 1D pooling are used in time series analysis (or 2D in the case of multivariate datasets). Figure 6 shows a univariate example.

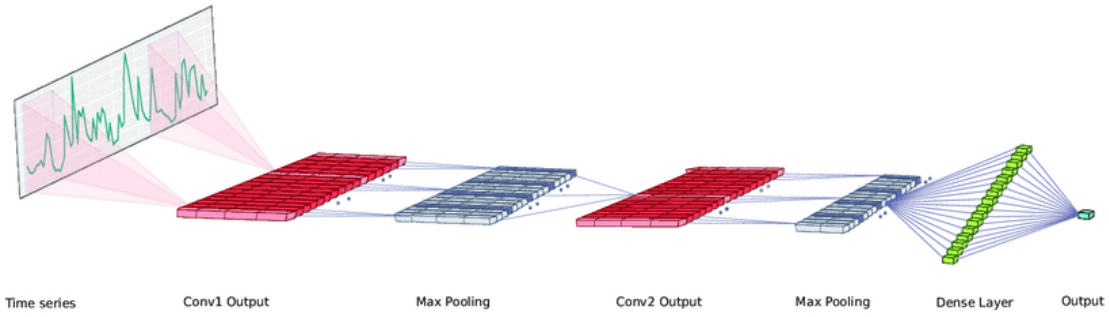


Figure 6: CNN for Time Series Prediction [13] [25].

By training the CNN, the system learns a data model for the time series and is able to make classifications or predictions following the dense output layers. The input samples are time series windows of a defined length $\{t_0, t_1, \dots, t_n\}$ and the target output is taken from subsequent time series measurements $\{t_{n+1}, t_{n+2}, \dots\}$. They are essentially supervised by the next window. Though the Figure above only shows a single output node, multiple predictions can be made by increasing the number of neurons at the output layer. CNNs also have fixed receptive fields based on their first convolutional layers.

CNN extensions include residual neural networks (ResNet) and WaveNet architectures. ResNets use skip connectors and *residual blocks* developed by He, et al. [31] to classify time series and avoid vanishing gradient problems associated with deep CNNs. WaveNet, by Oord, et al. [29] was developed for raw audio waveforms using dilated convolutional layers by applying a kernel to data larger than the filter size–skipping some

input values per stride step. WaveNets are able to learn long- and short-term features compared to basic CNNs which focus on local patterns. Braei and Wagner [13] applies both of these to time series modeling to make time series predictions.

2.4.2.1.3 Recurrent Neural Networks (RNN)

Though CNNs have been applied to sequential time series problems, its roots are in static samples such as images. For sequential forms of data–video frames, text, music– recurrent neural networks (RNNs) architectures are used [14].

Until now, all of the ANN architectures have had activations that occur in one direction, called *feedforward* networks. RNNs use a feedback mechanism in which a neuron produces an output, retains an internal state, and sends that output back into itself. At each time t, the recurrent neurons receive the inputs from the previous layer and the recurrent neurons' output from step $t - 1$. Looking at the RNN as a function over time is a process called *unrolling the network through time*. This process is shown in Figure 7.

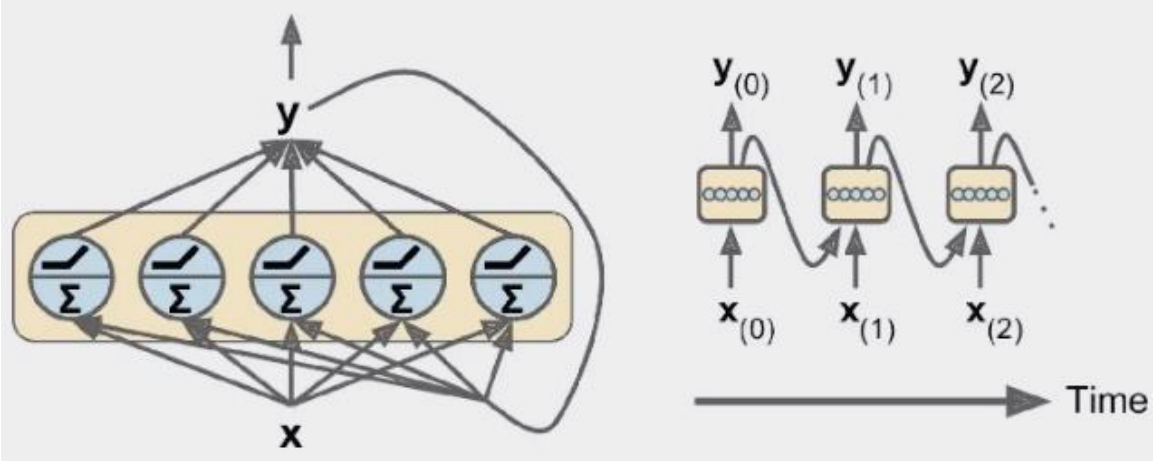


Figure 7: Recurrent Neural Network (RNN) Architecture [19, p. 499].

Feedforward networks have a fixed size input and output [14] which may have limited utility for processes that are not strictly held to a fixed width–like time series. With their feedback mechanisms, RNNs were specifically designed for this kind of arbitrary length data with specific applications in satellite monitoring [10] and emotion recognition [32]. Like CNNs, they are also able to include multiple hidden layers and multivariate inputs. However, RNNs suffer from unstable gradients which may never converge to a solution and are only able to remember short term behaviors [19, p. 497]. Similarly, *backpropagation through time* during RNN training can lead to exceedingly small gradients, *vanishing gradients*, as the loss is propagated to earlier time steps.

A common extension to the RNN includes long short-term memory (LSTM) cells. LSTMs reuse two vectors: h_t is added with the new data x_t, making it a short term memory, and c_t is multiplied with the new value, making it a long term memory. LSTMs use three gates–the forget, the input, and the output–to maintain longer term dependencies between significant events, determine what parts of the long term memory should be forgotten, and specify which part of the new long term memory should be sent to the output. This solves both the memory issues of RNNs and the vanishing gradient problem (especially the forget gate), but this is at the cost of longer training times [13].

Developed by Cho, et al. [33], gated recurrent units (GRUs) are a simplified version of the LSTM which compresses the three gates into one–the forget gate. Research has shown that the GRU's simpler design reduces computational complexity without a significant loss of performance [13].

Figure 8 shows basic LSTM and GRU cells.

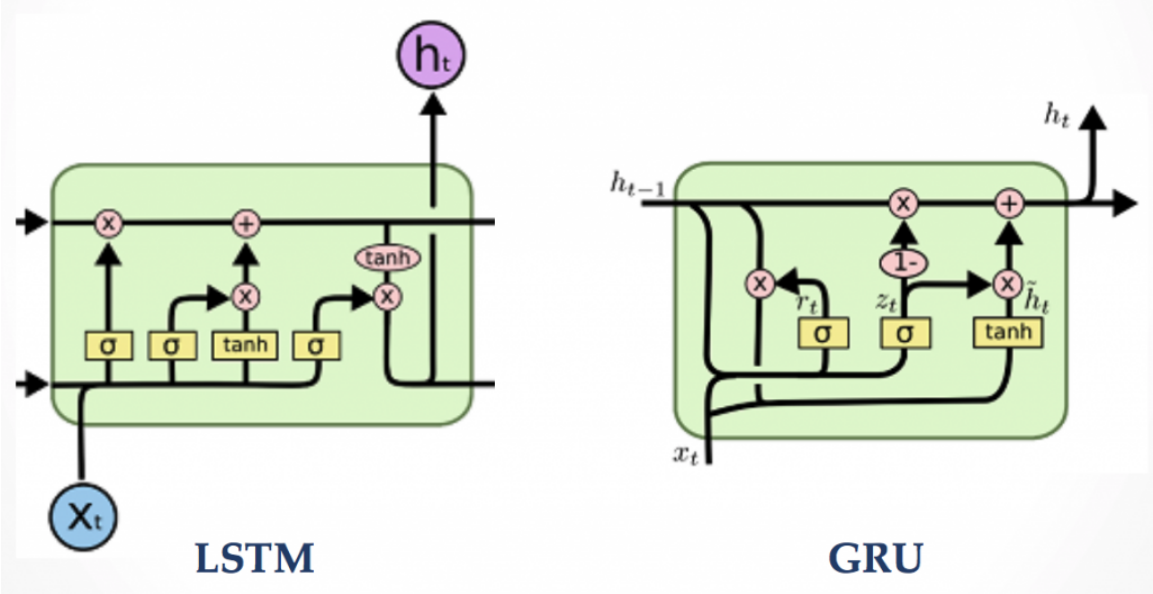


Figure 8: Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) Cells for Improving RNN Architectures [34].

RNNs take longer to train in general. However, they show excellent performance when applied to time series prediction, forecasting, and analysis tasks.

2.4.2.1.4 Autoencoders (AE)

Another method to create data models, the autoencoder (AE), reduces the dimensionality of the data and projects it to a lower dimensional space, where only the most correlated variables remain. When this lower dimensionality, or *latent space*, is projected back to its original size, only the most representative behaviors are rebuilt into the output. The compressing portion of the AE is called the *encoder* and the expanding portion is called the *decoder*. The two sides are made up of hidden layers, typically symmetrical about the latent space layer in the middle, and can share weights to accelerate training. Figure 9 shows an example of a basic autoencoder framework.

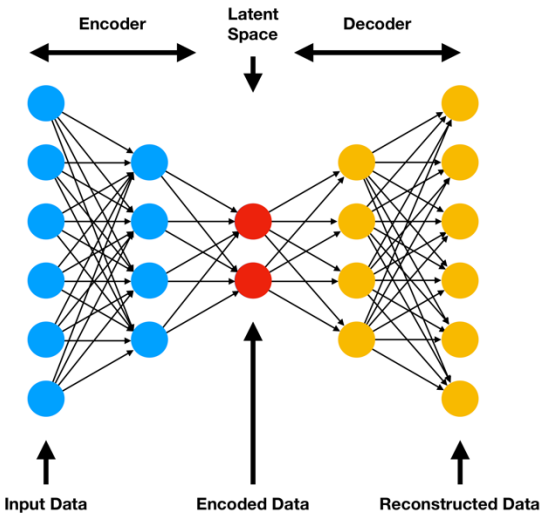


Figure 9: Autoencoder Architecture [35].

Essentially, these AE architectures are trying to reconstruct the same data as the input after passing it through a "bottleneck" of a lower dimensional representation. If $\boldsymbol{X}$ is the dataset, ψ is the decoding function, and ϕ is the encoding function, then the output would be expressed as Equation 21.

$$\hat{X} = \psi(\phi(X)) \tag{21}$$

With an optimization function attempting to minimize the difference between the $\boldsymbol{X}$ and the predicted $\widehat{\boldsymbol{X}}$:

$$\min_{\theta_\psi, \theta_\phi} \left|\left|X - \hat{X}\right|\right|_2 = \min_{\theta_\psi, \theta_\phi} \left|\left|X - \psi(\phi(X))\right|\right|_2 \tag{22}$$

where θ_ψ, θ_ϕ are the weights of the encoder and decoder, respectively.

MLP, CNN, and RNN architectures can be used as a framework for an AE with their accompanying strengths and weaknesses (including multivariate performance). When applied to time series, the same CNN or RNN layers are built with sequentially lowered

data structure shapes, then symmetrically rebuilt after the latent space, with the output having the same dimensionality as the input.

Autoencoders are typically trained solely with "normal" data, allowing the AE to learn only what is representative of usual behavior. If a dataset contains both normal and anomaly (or event) data, preprocessing based on clustering (discussed later) can perform an initial filter of high and low density patterns/behaviors to segregate out normal data [27]. With enough and a low enough percentage of outliers, anomalous or event data may be trained out of the data model since only the most efficient latent space is retained. This is the basis of *denoising autoencoders*, which applies stochastic noise to the inputs so that the AE will learn how to remove the noise.

Another extension of the AE, are *variational autoencoders*. Introduced by Kingma and Welling [19, pp. 586-590], VAE latent space layers also learn a mean and standard deviation coding. This is also a *generative autoencoder* capable of creating new instances that look like they were sampled from the training set.

2.4.2.1.5 Attention-based Transformers

The final system for data modeling, with specific application to time series data, is based off of the *attention mechanism*. Attention is a component for neural networks which learns which parts of an input sample is most influential in predicting another part of the output–a classification or prediction of a static tensor or sequence [19, p. 526]. The general attention mechanism consists of three matrix components: queries (Q), keys (K), and values (V). To compute attention [36], the query vector is matched against the database of keys to create a score value:

$$e_{q,k_i} = q \cdot k_i \quad (23)$$

Then, the scores are passed through a softmax:

$$\alpha_{q,k_i} = softmax\ (e_q, k_i) \quad (24)$$

Finally, the total attention is a weighted sum of the value vectors:

$$attention(q, K, V) = \sum_i \alpha_{q,k_i} v_{k_i} \quad (25)$$

This assigns higher weight and higher scores to more important features. Attention offers some explainability to the output of an ANN, especially *visual attention* for image classification. However, some researchers are skeptical that attention is truly interpretable [37].

Attention can boost the performance of RNN-based autoencoders, but it was the ground-breaking paper "Attention Is All You Need" [38] which significantly increased the utility attention by introducing the *transformer* [19, pp. 554-563], show in Figure 10.

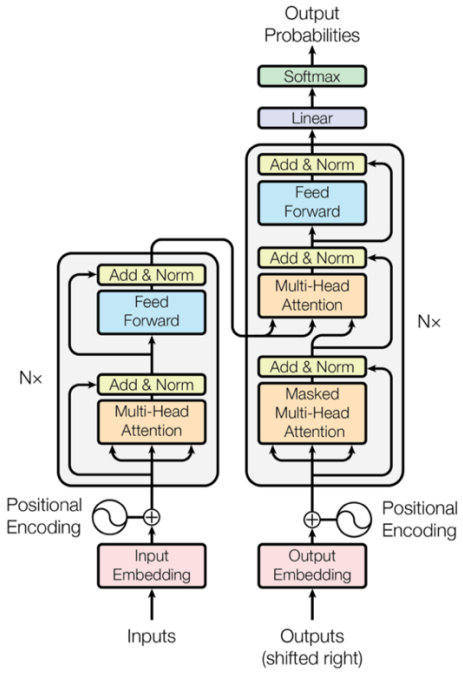


Figure 10: Transformer Architecture [38].

This learning architecture uses input and output positional embeddings or encoding, coupled with self-attention (plus dense and normalization layers) to create a robust, efficient sequence-to-sequence model. Since the transformer only uses attention, without recurrent or convolutional layers, the architecture is based mostly on simple matrix multiplication which has lower computational complexity and is parallelizable. This leads to lower training and inference time. The last component is called *multi-headed attention* which includes multiple self-attention modules and allows the transformer to learn more than one attention relationship for a single data input. In each of the attention layers, the matrices are usually symmetric. This represents a two-way causal relationship among the query and keys similar to the bi-directionality of RNNs. For time series, this may or may not be a desirable relationship. By masking half of the square matrix, attention only has a one-way causality where a current prediction only looks at preceding measurements [39].

Transformers are most prevalent in natural-language processing applications with recent advances with 3D human motion prediction [40], cross-dimensional geo-tagged measurement imputation [41], and video recognition [42]. However, time series are sequence models and only need an effective input embedding which encodes time to be effective. Time2Vec [43] is an example of this time representation which is invariant to time rescaling and can capture both periodic and non-periodic activity.

$$\boldsymbol{t2v}(\tau)[i] = \begin{cases} \omega_i\tau + \phi_i, & if\ i = 0 \\ F(\omega_i\tau + \phi_i), & if\ 1 \leq i \leq k \end{cases} \quad 26$$

where $\boldsymbol{t2v}(\tau)$ is a vector of size $k+1$, F is a period activation function, and ω_i, ϕ_i are learnable parameters.

Using Time2Vec, a transformer is able to learn a data model which can predict the next time-offset data series window from a previous sample.

2.4.3 Over- and Under-Fitting Models

In any method where a model is iteratively learned from training examples, there is a danger that the model will not perform well on new data. If the model learns to represent the original data too well, called *overfitting*, then the system will not be generalizable to instances that are not in the training. Conversely, *underfitting* the data typically means that the trained method was not sufficiently complex or applicable to accurately learn to represent the data [19, pp. 27-29]. Both represent potential issues when attempting follow-on event detections.

2.4.4 Considerations for Large Time Series

For large time series, long files with many time samples and/or multivariate sets with many inputs, many computers may not have resources to process an entire time series as one training or inference input. A long data set with thousands of samples may require many convolutional layers for a CNN and exacerbate the vanishing gradient problem of RNNs. Additionally, overfitting the training data may be an even greater concern with fewer, larger training examples. In many cases, a sliding window is used to create many shorter, often overlapping, 1D (or multivariate 2D) time series segments which can be trained in batches. More memory savings can be realized by reducing the overlap of the segments, similar to the convolutional stride. Using this windowed approach does create

additional hyperparameters of window size w and stride s, however it may make previously untenable research projects possible.

2.5 Unsupervised Event Detection Methods

Anomaly detection, or event detection in this time series application, starts with determining a model of "normal" data, then applying new test data to that model, and calculating a measure of dissimilarity between the expected data $\boldsymbol{X}$ and the model output $\widehat{\boldsymbol{X}}$. If that difference exceeds a threshold, an event is flagged.

Formally and in the general case, let w be the width of a data window, $\boldsymbol{X}$ is the dataset input, and the model function is ψ leading to a model output of $\widehat{\boldsymbol{X}}$.

$$\psi: \mathbb{R}^w \rightarrow \mathbb{R}$$

$$\hat{x}_i = \psi\big((x_{i-w}, \dots, x_{i-1})\big) \tag{27}$$

The difference score as a function of X, ϕ, is then calculated using the dissimilarity or distance function.

$$\phi_{score}(x) = d(x_i, \hat{x}_i) \tag{28}$$

where d is the dissimilarity function.

Lastly, this score is classified as an event if it exceeds a threshold, δ .

$$\phi(x) \longmapsto \begin{cases} event, \ if\ \phi_{score}(x) > \delta \\ normal, \ otherwise \end{cases} \tag{29}$$

The equation above represents a simple static threshold. This threshold could be selected using *a priori* domain knowledge (e.g. a low-voltage warning for a battery) or calculated using the training data. These thresholds can be for each single, univariate trace

in a multivariate set, or the multiple difference scores can be aggregated into a single metric and thresholded together. Additionally, dynamic thresholding methods could also be used and the target level could change as the binary classifier retains some historical state information and adjusts the threshold accordingly. An example from Hundman, et al. [44] detects spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. However, static thresholds are still common in event detection literature.

Supervised event detection algorithms that directly classify the event flag, such as a CNN with a logistic activated output layer, typically generate the best performance [14]. Classical machine learning methods, such as k-nearest neighbor classification, also do a good job of segregating dense clusters of normal activity and outliers [15]. In these cases, the system is essentially shown what an event is supposed to look like during training. In a limited domain like CPS analysis, where pre-labeled data is unlikely to exist, developing training samples could be prohibitively costly, the collection timeline may be too slow to make timely assessments, and the labeling may represent enough of the analysis task that injecting any autonomy would be redundant at best.

Sophisticated unsupervised methods, capable of drawing correlated behavior from multivariate data, are required to remove more of the labor-intensive manual processes, develop autonomous event detection, and accelerate decision-making.

Goldstein and Uchida [15] published a large-scale survey of unsupervised anomaly detection algorithms for multivariate data. The authors developed a taxonomy of four main unsupervised anomaly detection groups based on: nearest-neighbor (global and local), clustering based (global and local), statistical, and subspace. However, this thesis focuses instead on four categories: statistical, error-based, distance-based, and subspace-based.

Generally, these take a time series data model, either assumed or learned, then applies a dissimilarity function for static thresholding and classification.

2.5.1 Statistical Event Detection Methods

In event detection, statistical approaches still make assumptions about the underlying generative process but differ from time series modeling by then calculating statistics about the time series to determine a difference score.

2.5.1.1 Changepoint Detection Algorithms

Changepoints are statistically observable changes in the underlying model of a signal or time series. Truong, et al. [17] published a selective review of offline changepoint detection algorithms and defines three components of a detection algorithm:

1. Cost function - measures “homogeneity” and encodes the types of changes that can be detected.
2. Search method - the procedure for discrete optimization, exactly or approximately, associated with finding the correct number of changepoints.
3. Constraint - a penalty added to the optimization problem related to the amplitude of changes to detect.

When the number of changepoints is unknown, this problem could be viewed as an unsupervised method though the form of the generative process is assumed and no learning actually takes place. There are a host of cost and search options discussed in [17]. Online methods also exist. Each comes with strengths and weakness, however Lee, et al. [11] discuss two main weaknesses for changepoint detection algorithms in general. First, they

rely on prior assumptions of the parametric model of the time series which may be incorrect or violated by more complex time series. Secondly, they often utilize simple features extracted from the input data so only "statistically-detectable boundaries" can be located. The constraint especially shapes the magnitude of changes detectable with small penalties possibly being over-segmented by noise and large penalties only finding the most significant changes [17]. Slow changes in the simple features may be similarly missed. Lastly, changepoint detectors are univariate, therefore requiring a separate process for each signal and a novel aggregation method.

All of these options represent a large decision space with significant trade-offs between accuracy, computational complexity, and the types of changes detectable. A single method would likely be incapable of expressing all of the necessary complexity of this domain.

2.5.1.2 Histogram-based Outlier Score (HBOS)

The histogram-based outlier score [45] assumes the features are independent, then creates a histogram for each. For every instance in the dataset, the inverse height of the bins it resides in (essentially the density estimation) for all features are multiplied. There are two different binning modes:

1. Static bin sizes with fixed width.
2. Dynamic bins such that it maintains the same number of bins.

The dynamic method allows the density estimation to be more robust in the presence of large outlying values.

HBOS is computationally efficient but does assume independence between the feature traces.

2.5.2 Error-based Algorithms

These second methods are likely the most common for event detection and are based on an error metric.

2.5.2.1 Decomposition Residual Thresholding

Section 2.4.1.1 discusses the method of extracting the cycle, trend, and season from a time series that exhibits those characteristics. What remains is the residual signal, sometimes called the error. In its raw form, this residual will often appear noisy with measurements above and below a mean. It is common to normalize this sequence to vary around zero, representing over- and under-estimation based on the decomposition. In this case, a different positive and negative threshold could be set based on the user's application. Alternatively, the absolute value of the error can be used with a single threshold. Any residual which exceeds the thresholds are considered events [44]. Figure 11 shows an example of time series decomposition with thresholds.

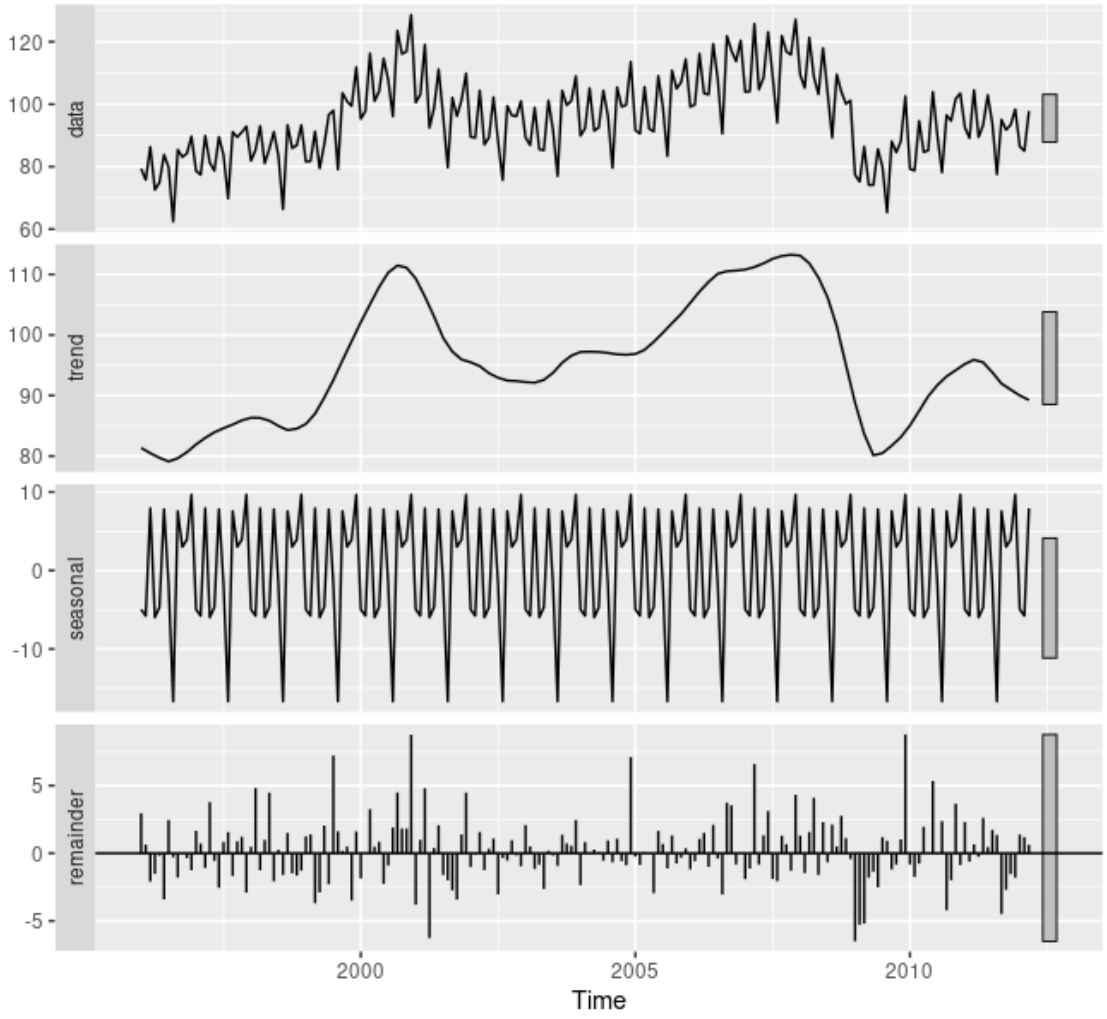


Figure 11: Time Series Decomposition [44].

2.5.2.2 Model Output Error

The second error-based detection method is common among ANN applications. These methods subtracts the actual data X from the output of a time series model $\hat{X}$. Any of the loss functions for the ANN training could be used, however a simpler metric is a simple difference.

$$\phi_{score}(X) = |\hat{X} - X| \quad 30$$

There are two general situations where this is used: forecasting and reconstruction. In time series forecasting , the model output is a subsequent, predicted timestamp or series of predictions. The actual measurements, likely from the next collected window, would be the actual data subtracted from the model output. In this case, the errors come from either incorrect predictions from an imperfect data model or as a result of an event. The reconstruction, involves output from an autoencoder architecture which is attempting to rebuild the input data after first reducing the dimensionality. The latent space is the most efficient model of the training data, so errors are created in the output when the input is not sufficiently represented.

This gives an error metric for each timestamp and, like residual thresholding, it is not uncommon to use the absolute value of the error. Again, this can be calculated individually for each data trace or aggregated into a system-wide metric. Regardless, an event is considered detected if any of the errors exceed a set threshold.

2.5.3 Distance-based Algorithms

Distance-based algorithms are related to error-based methods because they both calculate a metric of dissimilarity, however distances are created by inherent features of the input and not from an inability to perfectly represent the time series.

2.5.3.1 Clustering

Some classical machine learning techniques for determining outliers, and therefore events, use clustering mechanisms. There is no universal definition for a cluster, they are context and application dependent, and different methods group points differently [19, pp.

235-238]. In general, these methods minimize the distance between similar feature sets and maximize dissimilarity. Once the clusters are defined, the centroids are calculated and any new data is measured from these points. If the distance is greater than a threshold, that data point is a part of an event.

There are many different clustering methods referenced by Goldstein and Uchida [15] in their survey. K-means [19, pp. 238-249] is one of the more common. Some nearest-neighbors algorithms like k-NN which finds similarities globally and local outlier factor (LOF) [46] which act locally. Other strictly clustering algorithms include the global cluster-based local outlier factor (CBLOF) [47] and clustering-based multivariate gaussian outlier score (CMGOS), another local method [48] [49].

Clustering can also be used for pre-training to segregate high-density clusters of data points, likely to be normal time series behaviors, from samples in lower-density regions, which may represent events. Label-propagation methods can mark non-events in the dense regions which can then be used in supervised learning models [19, pp. 251-255].

2.5.3.2 Latent Space Distance

Lee, et al. [11] describe many problems with changepoint detection algorithms and propose a novel concept called breakpoints. This method for finding breakpoints trains a CNN autoencoder then segments a multivariate time series using latent space distance. In the case of event detection, the target detection is found at these segmentation transitions. Figure 12 depicts this method.

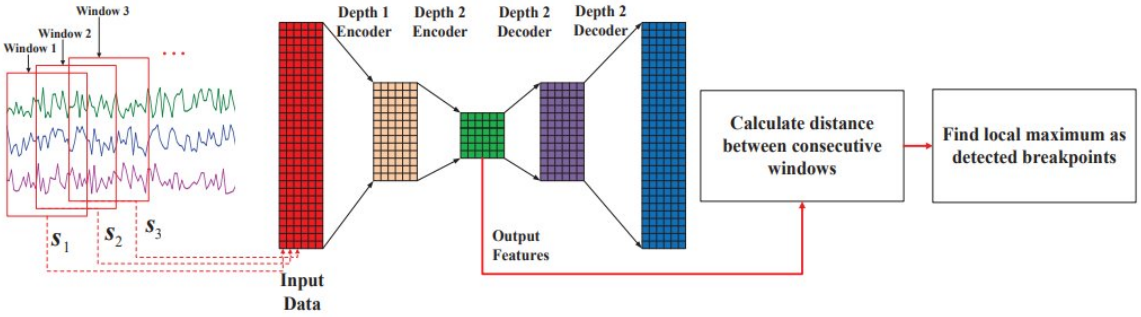


Figure 12: Event Detection in Time Series Using Latent Space Distance [11].

Autoencoders automatically learn the most relevant features to represent a data model from the training samples. When data is passed back through the AE, there is a unique activation matrix proportional to time series activity that is created in the latent space. To determine if the t^{th} timestamp is part of an event, this method calculates a normalized Euclidean distance (L2-distance) between the f_t and f_{t-1} latent space feature sets.

$$dist_t = \frac{||f_t - f_{t-1}||_2}{\sqrt{||f_t||_2 \times ||f_{t-1}||_2}} \quad 31$$

As a sliding data window approaches more abnormal activity, a possible event, the latent spaces are more dissimilar creating larger distances. The authors define breakpoints as local maxima in this distance space, but any distance that exceeds a set threshold could be considered part of an event.

One of the drawbacks of this method is attribution to an individual trace since it is impossible to know which portions of the latent space are from which input signal. This method can only be used for an aggregate, system-wide detection.

2.5.3.3 First- and Second-Differences

One of the methods for removing season in a classic decomposition is by subtracting the value of timestamp t from $t + 1$, called differencing. Depending on the complexity of the season, the time series may need to be differenced multiple times. This can also be a simple, computationally efficient way to model derivation. Therefore, a first-difference is similar to the velocity of a measurement and a second-difference is akin to the acceleration. This method can be used for any error-based metric or for the latent space distance mentioned above. A simple threshold is applied to either of these differences to detect an event.

2.5.4 Subspace Algorithms

Subspaces are lower-dimensional representations of a dataset. Principal component analysis (PCA) is a common analysis technique for finding these subspaces and, like other methods, significant deviations from normal subspaces may indicate anomalies or events. CMGOS mentioned above and a robust method for PCA (rPCA) [50] are examples of this subspace analysis [15].

2.5.5 Ensemble Methods

Like any classification or regression task, performance may be improved by ensemble methods, i.e. using a combined answer from multiple decoupled algorithms. A simple and common method is voting [19, pp. 189-192]. For event detections, at time t_n multiple methods are used which give an event or non-event classification. If enough of

the independent methods flag the behavior as an event, then the overall system determines that t_n is an event.

There have been some research examples for ensemble methods for anomaly or event detection [51] [52].

2.6 Measuring Performance

During a quantitative study, it is important to have objective metrics to compare empirical results. Like training the machine learning models, these evaluation metrics are used as the comparative benchmarks to judge the accuracy and cost (mostly in time) of the event detection methods.

2.6.1 Boolean Algebra

Event labels, either prepopulated to support supervised learning or generated through a detection method, are inherently binary. Equation 29 shows how a function applied to the target dataset will convert a continuous value/measurement into an event or not event. Even if a system were to use a multilabel classifier, the types of events to be marked are still a series of yes or no questions, true or false. In this binary domain there are additional mathematical operations called conjunction, disjunction, or negation. In computing, these basic logical operators are more often called *AND*, *OR*, and *NOT*. Additionally, there is an exclusive OR (XOR) function to determine whether one and only one of the binary inputs are true (or false). Table 1, called a truth table, shows how these binary operators work [53, p. 12] .

Table 1: Binary Truth Tables.

Logical State		Operation		
Binary Value		**NOT**		
TRUE		FALSE		
1		0		
FALSE		TRUE		
0		1		
		AND	**OR**	**XOR**
TRUE	TRUE	TRUE	TRUE	FALSE
1	1	1	1	0
TRUE	FALSE	FALSE	TRUE	TRUE
1	0	0	1	1
FALSE	TRUE	FALSE	TRUE	TRUE
0	1	0	1	1
FALSE	FALSE	FALSE	FALSE	FALSE
0	0	0	0	0

Boolean algebra extends beyond these basic operators to conditional statements, logical equivalencies, propositional logic, and DeMorgan's Laws [53]. However, these topics are not used in this thesis.

2.6.2 Evaluation Metrics.

When the event detection algorithm flags a data point as an event, the system can be correct or wrong. To make this type of qualitative evaluation, there must be a target to compare with the system output. Typically, this requires annotations, or labels, generated by a human with a level of domain expertise. However, this could also be comparing the outputs of two different detection algorithms.

In most cases, the detection system is only allowed to make an event or not event output. When compared against a set of evaluations considered "truth," the system assessment can be correctly true (true positive, TP), correctly false (true negative, TN),

label a truth that is actually false (Type 1 error, false positive, FP), or label a false that should be a true (Type 2 error, false negative, FN) [54]. Figure 13 visually shows these relationships.

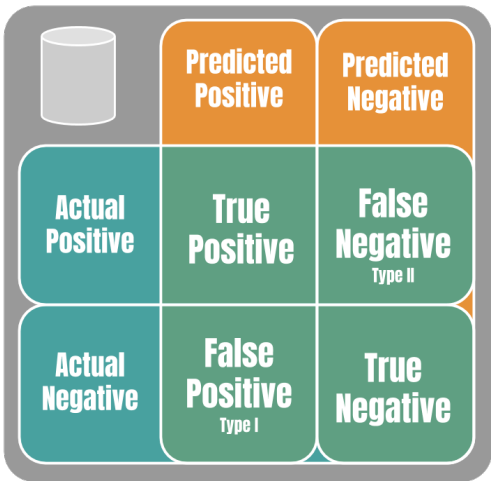


Figure 13: Binary Classification Confusion Matrix [54].

The most basic evaluations for this type of classification system are accuracy, precision, and recall.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad 32$$

$$Precision = \frac{TP}{TP+FP} \quad 33$$

$$Recall = \frac{TP}{TP+FN} \quad 34$$

The percentages, or rates, of the correct and mislabeled data are also considered.

$$True\ Positive\ Rate: TPR = \frac{TP}{TP+FN} \quad 35$$

$$False\ Positive\ Rate: FPR = \frac{FP}{FP+TN} \quad 36$$

Specific metric to the evaluation of event detection include the annotation error, Hausdorff distance, F1-score, and the area-under-the-curve of the receiver-operator characteristic curve.

2.6.2.1 Annotation Error

The annotation error is simply the difference between the number of true events and predicted events. Especially when the true number of anomalies/outliers/events are unknown (as in the case of unsupervised learning), this calculation is a simple metric for comparing the accuracy, precision, and consistency between individual expert annotations and those annotations with autonomous model detections. Ideally, these numbers are identical and the annotation error is 0.

$$AnnotationError \coloneqq |\widehat{K} - K^*| \tag{37}$$

where $\widehat{K}$ is the predicted number of events and K^* are the true number of events.

2.6.2.2 Hausdorff Distance

The Hausdorff distance measures the space between two subsets of a metric space. From [17], this measures the robustness of a detection method or the greatest temporal distance between a change point and its prediction. It is measured as the number of signal samples between an "event" detected or annotated and the identified "truth" value. If the metric is 0, then the two markers are identical.

$$Hausdorff\left(T^*, \widehat{T}\right) \coloneqq \max\left\{\max_{\hat{t}\in\widehat{T}} \min_{t^*\in T^*} |\hat{t} - t^*|, \max_{t^*\in T^*} \min_{\hat{t}\in\widehat{T}} |\hat{t} - t^*|\right\} \tag{38}$$

where $T^*, \widehat{T}$ are the time lengths of the true and predicted times between events and $t^*, \hat{t}$ are the corresponding timestamps of the true and predicted events.

2.6.2.3 F1-Score

F1-score is an evaluation metric which evenly balances precision (the fraction of true positives retrieved versus all flagged data) and recall/sensitivity (the fraction of true positives retrieved versus all actual positives).

$$F_1 = 2\frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \quad 39$$

Precision, recall, and F1-score all vary between 0 and 1. Precision is closest to 0 and recall is closest to 1 when all events are detected, typically by setting the detection threshold arbitrarily high (i.e. all time samples are flagged as "events"). The reciprocal is true when fewer but more accurate detections are made, often at the cost of missing some detections.

2.6.2.4 Receiver-Operating Characteristic (ROC) Curve

The receiver operating characteristic (ROC) curve is a graphical representation which illustrates the diagnostic ability of a binary classifier. Typically associated with simple classification tasks, the ROC curve sweeps through the system's discrimination threshold and calculates the true positivity and false positivity rates at each level.

Though the ROC provides a graphical representation of the diagnostic ability of a classifier, it does not provide a metric to easily compare two classifiers. The area-under-the-curve (AUC) of the ROC curve is calculated to provide an objective measure.

An optimal classifier has an ROC curve which approaches the top left corner of the graph, maximizing TPR and minimizing FPR, leading to an AUC closer to 1. A classifier approaching random, closer to 50/50 odds, has a ROC which is a diagonal between 0 TPR/0 FPR and 1 TPR/1 FPR (AUC close to 0.5). An AUC close to 0, and an ROC approaching

the bottom right corner, represents a misclassifier which always, but accurately, predicts the wrong class.

2.6.3 Time Metrics

Time represents one of the most costly resources when humans are involved. Creating sufficient labeled data for supervised training methods is typically one of the most time and resource costly parts of a machine learning project. Even statistical methods require the time-consuming annotations to use as the objective comparison to judge performance. Lastly, it is necessary to know how much time an autonomous system saves (or adds to) domain experts to judge the value of an automated process.

There are five times to be considered in this thesis.

2.6.3.1 Collection Time

The first time is the required duration to collect data for analysis. For online methods, this could be a function of the sampling rate of a signal and the number of data points required for the next event detection to make an assessment. However, online methods would probably only be used after an end-to-end event detection pipeline has been identified and tested.

Offline methods, which are the focus of this thesis, would use collection time as the time it takes to collect a data file. Most domain applications likely use real time collection, i.e. logging the telemetry from the system as it is performing an activity. The data would be saved and retrospectively analyzed for creating a temporal data model and event detection analysis. Depending on the lengths of collection sessions, this could represent a

significant time investment necessary to generate enough data to be sufficiently representative of detection targets. It may also require human operators to actually perform the collected activity. However, some domains may be able to use simulation data for some or all initial training and early validation. This could occur in a digital environment, much faster than real time and would not require significant man-hours of effort.

2.6.3.2 Annotation Time

Annotation time is how long an annotator, also called a coder, takes to label a data file–a multivariate time series in this case. The time would be dependent on complexity of the data, annotator familiarity with the dataset, domain expertise, and also could simply be a function of the intuitiveness or ease of annotation software. Annotation time represents the cost to perform the event detection without autonomous intervention. It is an important baseline when evaluating the time savings of an autonomous detection system.

2.6.3.3 Training Time

Machine learning methods take time to train themselves to draw inference, and generate a model, from the input data. Training time includes this upfront cost required before an event detection technique can be used. Generally, faster training times are desirable, however increased performance may be worth the cost of additional training cycles. Training can luckily be done offline, i.e. without a human needing to have active interactions with the system. The training can be done outside of work hours, or in the background, while an analyst is performing other tasks. This time could also include how long retraining takes to accommodate new data.

Training may take a long time–hours, days, even much longer in some applications. For example, one of the cutting edge deep learning language models, BERT, uses deep bidirectional transformers for learning understanding and is known for taking a long time to train—even on extremely powerful computational hardware. Only recently has the training on 16 TPUv3 chips been reduced from 3 days to 76 minutes [55]. However, it may not matter how long this training takes as long as the model is initially generated or updated with a fast enough cadence that the results are sufficiently timely to be useful.

2.6.3.4 Inference Time with Event Extraction

Inference time with event extraction is when the detection method is actually run. This is run after a machine learning algorithm has completed its training. For some statistical methods, this could be the time . In both cases, this time includes both how long the system takes to generate the metric series, e.g. reconstruction error, and the time required to detect events against that series.

Inference time could also be run outside of work hours but may be run with a human actively waiting for output. Therefore, short inference time may be a stricter requirement than training time.

2.6.3.5 Correction Time (Human Acceptance)

Lastly, the time required for a human to correct the outputs of the event detection and accept the final result is called the correction time. The same dimensions of data familiarity, domain expertise, and tools as the annotation task apply here as well. Coupled with the inference time, this combined event detection time can be compared against the

annotation time to determine actual time savings (or lack of savings) associated with the autonomous system.

Once the accuracy, reliability, and other performance metrics for a system are well established, a human may no longer be required to correct every output. The system could transition to a more spot check or maintenance model to ensure the continued performance. However, this is very domain dependent. The amount of trust a developer has in the system may not be the only constraint. Policy, law, and the spectrum of costs for errors (e.g. monetary or safety of life) all contribute to the requirements for correction.

2.7 Method Search Optimization

When comparing the performance of the above time series modeling and event detection methods, there are an enormous number of options to test. The particular changepoint detector components and penalties; learning rate, optimizers, and number of epochs in machine learning; and/or the architectures, depth, and width of ANNs are just a few examples of the hyperparameters that should be tested. Though some methods may perform better in general, specific optimization is likely unique to the domain, application, and possibly even target detection. Some search methods include: random search which test a random set of hyperparameters at each epoch [19, pp. 321-322]; grid search which systematically tests hyperparameters according to a specified testing regime [19, pp. 320-321]; and evolutionary search which randomly changes a certain number of hyperparameters and further tests the best performers [56]. Any optimization search is a compromise between speed and thoroughness. One way to balance this is by doing an

initial coarse search then refining. For EDAS, a single parameter sweep of pre-processing window size and model architectures.

2.8 Research Design

A core measure of success for this thesis is delivering value to an end user, likely in the form of time savings and improved detection accuracy. To evaluate this utility, a multi-phase mixed methods approach following a case study design of inquiry [57, p. 14]. An in-depth analysis of a particular domain, dataset, and analytic toolset will be supplemented by a pair of survey instruments.

Mixed methods research draws procedures from both qualitative and quantitative research. Quantitative research tests objective theories by examining the relationships between variables [57, p. 4]. Qualitative research is an approach for exploring meaning people apply to a human problem [57, p. 4]. Creswell [57, pp. 215-239] discusses how to perform research using mixed methods. He first defines three basic types of mixed methods –convergent parallel, explanatory sequential, and exploratory sequential–each with their own considerations for data collection, data analysis, interpretation, and establishing validity of the results.

This research starts with an exploratory study followed by a convergent parallel method.

Part 1 starts with a pre-survey to establish participants' demographics, experience, and initial perceptions of the manual and autonomous event detection problem. Part 1 also asks participants to hand-label data and time their activity. This establishes a baseline for

the manual task. Part 2 begins by asking the participant to run a selected detection algorithm which autonomously extracts events. Then, they are asked to validate and correct the outputs while also recording their time. This creates the dependent variable to objectively test process acceleration. Part 2 concludes with a post-survey which qualitatively assesses the participants' perceptions of the autonomy intervention.

Participants will be self-selected, domain expert volunteers from a small population and perform both annotation and survey tasks in their *natural setting* [57, p. 185].

2.8.1 Assessing Human Annotations

Humans are noisy when multiple people try to annotate the same phenomena. Experts may disagree when hand-labeling material. There may be "wisdom of the crowds," but cross-checking and validation is still critically important, especially in high cost domains. Creswell [57, p. 203] describes the importance of characterizing intercoder agreement during qualitative studies. Miles and Huberman recommend up to 80% consistency for good qualitative reliability [58].

In a domain as subjective as CPS event detection, it can be difficult to establish a truth version. However, methods such as Cohen's kappa (Equation 40) attempt to balance the agreement based on the number of possible selections [59, p. 162].

$$kappa = \frac{d-q}{N-q} \tag{40}$$

where d are the number of agreements, q is the sum of agreements by chance, and N are the number of choices.

Like many other statistical correlation metrics, Cohen's kappa varies [-1,1] where 1 is a complete agreement, 0 occurs when there is no more agreement than expected by

chance, and negative values when coders tend to give different ratings. According to McHugh [60], values ≤ 0 as indicating no agreement and 0.01–0.20 as none to slight, 0.21–0.40 as fair, 0.41– 0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1.00 as almost perfect agreement.

Other research includes “Integration of multiple annotators by aggregating experts and filtering novices” by Zhang and Obradovic [61] and “Aggregating and Learning From Multiple Annotators” by Paun and Simpson [62].

2.8.2 Survey Methods

Both surveys use Likert-like scales, which forces participants to select one of five answers that represent levels of “goodness”, will be used for statistical comparison. Additionally, open-ended short answer questions will be used and analyzed by finding and interpreting common themes.

2.9 Considerations for Follow-On Classification

The final consideration for the event detection task is how to prepare event data for the following classification task and active learning. The goals associated with Step 7 and 8 of the analytic process (Figure 1in the Introduction). The biggest contributor to this decision is what format the classifier requires as input data. Some neural network classifiers can directly use 1D time series (or 2D in multivariate cases). Others can use the data transformed into the frequency domain using Fourier transforms or use image recognition CNNs if the time series undergoes a wavelet transform [28]. In the case of

autoencoders, the event detection algorithm can even be used as pre-trained layers before a follow-on classifying architecture [19, pp. 576-579]. Like framing the event detection problem from Section 2.3, having an idea of how this step fits into the overall automation pipeline could save time down the road.

2.10 Summary

Event detection from time series data has a long history with many applications and domains. Time series event detection is closely connected with time series forecasting, with many techniques building a model to predict values and then calculating the prediction from actual measurements. There are two main categories for time series analysis: statistical methods which assume a generative model for the stochastic process and then measure deviations from that model; and machine learning models which learn a representation for the data model directly from the data and then measure deviations from the calculated model. Statistical methods include time series decomposition, AR, MA, ARMA, and ARIMA for time series modeling. Machine learning is further divided into classical methods, that have historically struggled with time series modeling, and artificial neural nets, which include convolutional-, recurrent-, and transformer-based models. After a time series model is created, error and/or distance metrics such as reconstruction errors, outlier scores, histogram binning, or latent space distances are calculated. Events are detected when these metrics exceed a set threshold that is between data considered normal and data considered a part of an event. Evaluation metrics of the event detection outputs included F1-score, annotation error, and Hausdorff distance.

This Chapter concluded with a discussion on qualitative case studies, metrics for inter-coder agreement (e.g. F1-score, Cohen's kappa, annotation error, and Hausdorff distance), and Likert-like scale plus short answer questions associated with participant surveys.

III. Event Detection Automation System

Event detection automation system (EDAS) is a novel detection system for cyber-physical systems (CPS) domains. This method uses deep-neural networks to accelerate the identification of *events* in time series data. Throughout this methodology, *annotations* will reference the baseline labels developed during Part 1, *detections* are those events which are output by the autonomous process, and *corrections* will be the expert corrected data from Part 2.

3.1 Data

As presented in Chapter 2, many event detection methods require a seasonality or trend, training data containing only "normal" data, and/or an extensive hyperparameter search that can change the detection criteria (possibly including or excluding events at each setting). If the analysis domain does not align with a program's required timelines or data suitability, the above techniques struggle to identify events effectively and efficiently.

This research uses time series data signals logged from the Controller Area Network (CAN) from a personally owned and operated vehicle (POV) and two drivers. The CAN traffic carries packetized data associated with all of the electrical components controlled by the vehicles' on-board computers. The CAN interface control document (ICD) [63] states that the packets contain one to eight byte-words with transmission priority based on the packet ID. The data includes both analog/continuous signals (e.g. engine RPMs, vehicle speed, and brake pressure) and discrete signals (e.g. single bit indicators for turn signal on/off status).

CAN data exhibits the necessary characteristics of the CPS domain of interest to be used for this research:

- No seasonality or overall trend
- Discrete and analog signals
- Steady-state and event behaviors without an establishable “normal”

The dataset includes 13 files logged from a 2017 Honda Civic. Each file represents “ambient driving” activities during normal visibility and driving conditions. Benign anomalies (e.g. opening a door while driving), hardware/software faults (e.g. tire blowout or cyber attack), and emergency maneuvers (e.g. anti-lock brake or airbag engagement) were not a part of the data collection.

A CANedge1 logger from CSS Electronics was used to collect the CAN data from the POV.

After the files were logged, this researcher used domain expertise and knowledge to perform initial data analysis and parameter identification. During the collections, 56 unique packet IDs were observed with approximately a 33.3 kilobits per second data rate. To simplify the data representation of the vehicle, seven traces were selected which represent most applicable activities. The parameters were then appropriately formatted and aligned into a time-tagged dataframe. Table 2 shows a summary of these parameters. The packet IDs are obfuscated to protect proprietary information. However, the parameters are grouped based on which packet they are transmitted, and the relative priority of each packet is preserved by the naming convention.

Table 2: Honda Civic Packet Data, Representative Signals.

Packet ID	Parameter	Format
1	steering position	16-bit 2's complement
2	vehicle speed	16-bit magnitude
3	accelerator position	8-bit magnitude
	engine RPMs	16-bit magnitude
4	transmission state	8-bit discrete
5	brake pressure	16-bit magnitude
6	turn signal	8-bit discrete

Additional preprocessing was used to accommodate different packet output rates and large differences in signal dynamic ranges created by the various binary formats. The raw outputs were min-max scaled to force all ranges between 0 and 1. This prepared the data to better utilize deep learning architectures and ensured that the training would not be dominated by the 16-bit numbers. The resulting dataframe was relatively sparse due to the different output frequencies of each packet. The final dataset linearly interpolates analog signals and right-fills the discrete parameters (i.e. the previous value is held until a new sample changes it). This is a valid method to fill the dataframe because the data sampling rate far exceeds the CPS's response to user, environmental, and CAN inputs. The data imputation should not be noticeable at the time scale of events but does prepare the data for ingestion by established deep-learning tools.

A sample of resultant time series data sets is shown in Appendix B.

Developing a general method to deal with different sampling rates, scaling factors, and nominal data is outside the scope and should be explored in Future Work.

3.2 Event Detection Automation System Overview

Figure 14 shows the autonomous event detection pipeline. There are multiple options available at each step in terms of the different pre-processing methods, time series modeling, and event detection techniques. This research, particular case study and dataset, the pre-processing methods and static thresholding were chosen *a priori*, so the algorithms being compared were the type and shape of artificial neural network architecture used in the model optimization search.

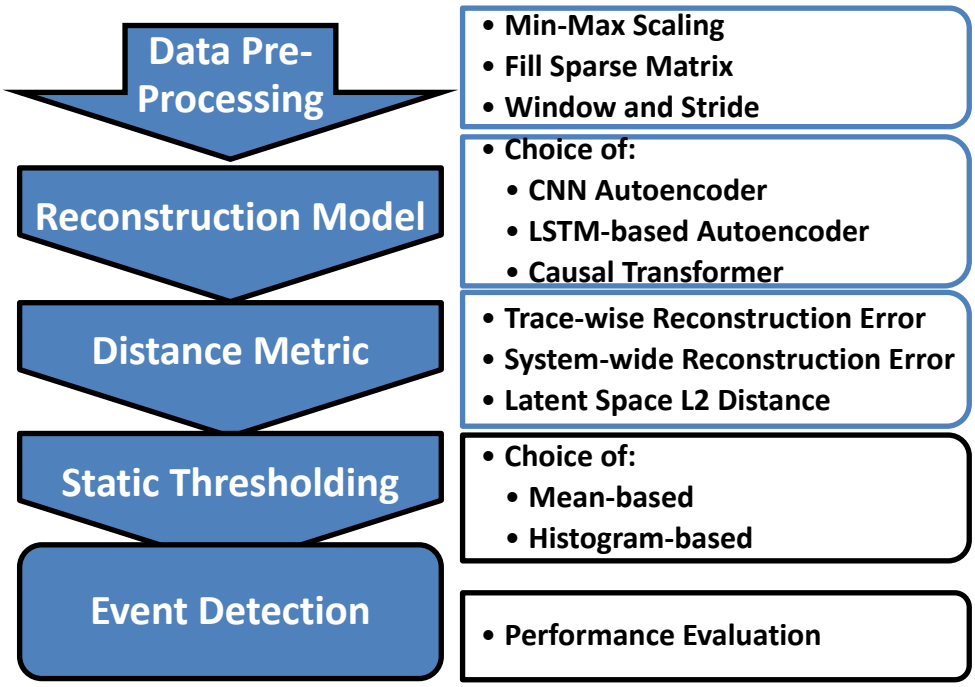


Figure 14: Autonomous Event Detection Pipeline.

The human participants only evaluated one of the reconstruction models and one distance metric and static threshold pairing. Both of these choices were for simplicity and explainability to case study participants who may not have much educational foundation in artificial intelligence or machine learning methods.

3.3 Data Pre-Processing

The seven channels of CAN data, the case study dataset, comes in several different dynamic ranges and formats. Similarly, the data is a single time series which is not suited for most of the time data model training methods. To optimize the data for the learning models, each trace is first scaled and then windowed.

3.3.1 Min-Max Scaling

The first step is to scale the data. On a per-file basis, each trace is min-max scaled using Equation 41.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad 41$$

where x_{scaled} is the scaled value for an input x and x_{min}, x_{max} are the lowest and highest values represented in a given trace. This was used to ensure that true zero values stay at zero since some scaling methods can move the lowest value.

This guarantees that each trace only varies between [0,1]. Many computational packages can save this scaling state and will be able to rescale the data after inference or prediction.

3.3.2 Windowing and Stride

Deep learning methods perform better on greater numbers of training samples. The CAN collections are large files that would be unfeasible to analyze as a whole. To accommodate batch learning, a sliding window with a given stride is passed over the time series and saved before training and inference. The window size and stride are

hyperparameters which determine the input shape and are identified with a parameter sweep during the coarse optimization of the detection method.

3.4 Reconstruction Model: Architecture, Training, and Lower-Dimensionality Representation

Though neural networks and attention-based models are better known in computer vision tasks and natural language processing, recent advances have demonstrated success in time-series forecasting, time-series analysis, and the associated anomaly/outlier/event detection task. As mentioned previously, Braei and Wagner [13] discuss several methods for anomaly detection to include deep-neural networks. However, their research focuses solely on univariate data sets with limited applicability for this multi-signal event detection problem. Additionally, research using time-series autoencoders have shown that relatively infrequent occurrences in the data can be observed when that same data is first reduced to a lower dimensional space.

During initial technique exploration performed by the researcher, deep learning methods were shown to have the greatest potential for time series modeling for datasets with CAN characteristics. The non-linear, complex relationships and the desire to establish cross-channel correlations in multivariate data drives this method toward sequence-to-sequence CNN, RNN, and transformer models.

After the dataset is prepared with a given window size and stride, that data is used to train a network which reduces the dimensionality to a latent space representation before attempting to reconstruct itself. This thesis uses three autoencoder-based network

architectures for sequence-to-sequence reconstruction of the vehicle data. Then, error and distance metrics between the original and reconstructed data are used to determine which data points belong to an event. Quantitatively, these methods are compared to determine the highest performance in this domain and dataset. The highest performer is used during the Part 2 portions of this research.

Two deep neural network approaches are discussed, a) a convolutional and LSTM-based recurrent neural network, and b) an attention-based network: the transformer. Part of this thesis involves human-interactive trust and a human-perceived reduction in the time to perform event detection tasks. Therefore, relatively basic and small architectures were selected to reduce model complexity and decrease training time. Nakkiran, et al. [64] also observed diminishing returns for deeper and wider autoencoder architectures. Each set of data window size, stride, and model architecture were trained using batch gradient descent with an ADAM optimizer. Table 3 summarizes the other hyperparameter selections that are common across each test regime. Additional hyperparameters and other architectures were beyond the scope of this research and should be explored as Future Work.

Table 3: Shared Model Training Hyperparameters.

Hyperparameter	Setting
Epochs	100
Batch Size	512
Learning Rate	0.001
Loss Function	mean absolute error
Train/Validate Split	0.9/0.1
β_1, β_2	0.9,0.999
ϵ	1.00E-07

3.4.1 Convolutional Autoencoder

The first method for data reconstruction is a deep CNN autoencoder. Compared to full-connected multi-layer perceptron, CNNs use convolutional layers–partially connected kernel windows–to reduce the number of trained parameters. CNNs have fixed memory, fixed size reconstruction capability and are generally used to find local patterns in data.

After the input layer of the neural network, CNNs typically have multiple convolutional layers each followed by pooling layers (for normalization and to reduce overfitting). For time-series data, a 1D convolutional layer (a 1xn kernel where n is the kernel size) is used. The number and depth of CNN layers typically vary between domain applications and datasets, but autoencoders require a reduction in hidden-layer size down to a "bottleneck," then symmetry between the *encoder* and *decoder* hidden-layers to allow the network to reconstruct the input data with the least error possible. Tested CNN architectures are described in Section 5.2.

This research uses relatively small, narrow CNN autoencoders with a set kernel size (k = 4), same padding, no convolutional striding, and tied weights between the *encoder* and *decoder*. All hidden layers use rectified linear unit (ReLu) activations, and the output layer uses a sigmoid activation to normalize the output to between 0 and 1 (to allow rescaling the output to the original input signal magnitudes). This method also does not use pooling between the convolutional layers due to the size of the networks and desired simplicity. Table 4 summarizes the hyperparameters shared by every CNN autoencoder.

Table 4: Convolutional Neural Net (CNN) Autoencoder Hyperparameters.

Hyperparameter	Setting
kernel size	4
kernel stride	1
padding	same
Conv1D activation	ReLu
Output activation	sigmoid

3.4.2 Long Short Term Memory (LSTM) Recurrent Autoencoder

The second data reconstruction method uses a recurrent neural network (RNN) autoencoder. RNNs have both feed-forward and feed-back mechanisms which allow the neural network to account for, and predict based on, data points before and after a given time point. This sequence-to-sequence architecture is a natural fit for time series data. Long Short Term Memory (LSTM) networks are an extension of basic recurrent neurons and can regulate the amount of data remembered or forgotten when the neuron's output is calculated. LSTMs dramatically reduce the amount of required memory to predict a data point at inference time for longer sequences. When compared to a CNN, LSTM-based RNNs are able to train on inputs of variable length and output arbitrary length sequences (of diminishing accuracy as the length exceeds the training data). RNN autoencoders, like CNN autoencoders, attempt to recreate the input sequence after reducing the data through a lower dimensional space.

After the input layer, this thesis uses LSTM-based RNN hidden layers to construct the autoencoder with 100 parallel LSTM units for both the *encoder* and *decoder*, hyperbolic tangent (tanh) activations, and sigmoid recurrent activations. Again, the output layer uses a sigmoid activation to ensure that the output varies between 0 and 1 to facilitate

the reversal of the min-max scaling. Lastly, the output layer also uses a time distribution function to recreate the multivariate traces from the LSTM decoder output. Though LSTMs can create variable length outputs during inference, the windowing of the data set during preprocessing means the RNN, like the CNN above, the LSTM predicts the same length output for every input. Table 5 summarizes the hyperparameters shared by each LSTM autoencoder.

Table 5: LSTM-based Autoencoder Hyperparameters.

Hyperparameter	Setting
LSTM layer activations	tanh
recurrent activations	sigmoid
output activation	sigmoid

3.4.3 Causal Transformer

The third deep learning architecture used in this thesis was a sequence-to-sequence transformer. For time series data, where RNNs sequentially calculate each individual time step based on previously predicted data points, the attention-based transformer supports parallel computing, and accelerates both training and inference, by simultaneously calculating every point in the series. Additionally, the positional encoding or embedding can support arbitrary length inputs and create variable length outputs (with significant degradation in accuracy after the output length exceeds training data).

This research uses a single-headed transformer with the Time2Vec [43] learnable embedding. The tested transformers also use a causal mask which ensures each prediction is only affected by the tokens preceding the current output (due to the time linear nature of POV signals). During training, the inputs and target outputs for the transformer are shifted

by one sample. This makes the transformer act similar to an autoencoder and is attempting to recreate the same data offset by only a single count (which corresponds to a small time difference). Similar to the LSTM, the transformer will predict the same length time series as the inputs despite being capable of variable length output (due to the fixed length input window).

3.5 Error and Distance Metrics

Once the representative data models are trained, either the autoencoders or transformer, the time series outputs are compared against the input data. This creates an error or distance space where each time sample has an associated metric to determine which samples are part of an event and which are considered steady-state activity. The resulting model output creates a data set of shape $n \times w \times c$ where n is the total number of time windows, w is the width of the window, and c are the number of channels. In each case below, the absolute value error or distance is summed over the 2nd dimension to create a $n \times c$ dataset where an individual time sample only has a single metric point. This accounts for both over- and under-predictions, maintains a positive value for every time step, and simplifies follow-on analysis by compressing the inference output to a 1-dimensional series per signal/channel.

To be discussed further below, an event is considered detected when the error or distance metric for a given time sample exceeds a static threshold.

3.5.1 Reconstruction Error

The first error space time series is created by simply calculating the difference between the model output's reconstructed series, $\hat{X}$, and the original input data, X.

$$error \coloneqq \epsilon_t = \sum_{i=0}^{w} \left| \hat{X}_i - X_i \right| \ \forall\, t \in T \tag{42}$$

where t is a given time step within the file length T.

This creates a time series of reconstruction errors. When an autoencoder or transformer is less capable of recreating the input data, and leads to greater reconstruction error, it is typically in response to a certain trace behavior being less present in the training data. The data is less representative of steady-state activity and more likely to be a part of an event. As the sliding window traverses data which contains a deviation from steady-state, additional samples in the window will exhibit more reconstruction error. When the window is summed/compressed after inference, the error metric will be maximized when the window is centered on an event–when the model is least able to reconstruct the data (likely associated with an event).

A second reconstruction metric was found by taking the first-difference of the reconstruction error.

$$diff \coloneqq \delta_t = \epsilon_t - \epsilon_{t-1} \forall t \in T - 1 \tag{43}$$

where ϵ is the reconstruction error at time t in the time series T.

This second error trace is equivalent to the rate, or velocity, of the change in error. Similar to the first metric time series, the first-difference measures the autoencoder's and transformer's inability to reconstruct the original input. However, this metric calculates

the speed at which the reconstruction error is deviating from the original trace. Higher first-difference values correspond to higher rates of change and likely precede or follow events.

Since autoencoders and transformers output in the same shape as the inputs, the reconstruction and first-difference errors metrics can apply to both individual traces and the system-wide aggregate traces. Event detections can occur in each data signal alone, as a group with other traces, or as part of the overall system. Since the system can find which signal is contributing the most error, this can allow for a more informative detection when presented to a follow-on autonomous system or human validator.

3.5.2 Latent Space L2 Distance - Autoencoders and Transformer Attentions

As published in [11], the bottleneck of a time series CNN autoencoder creates a compressed, latent space representation of the input data. After training and during inference, there is a unique two-dimensional "image" which is created–an aggregate, lower-dimensional representation for the whole system for a given window. By taking the L2 distance between each window, a $t \times c$ time series of matrix distances is created.

$$distance := d_t = \frac{||f_t - f_{t-1}||_2}{\sqrt{||f_t||_2 \times ||f_{t-1}||_2}} \quad 44$$

where f_t is the latent space feature vector at time t.

Though [11] only discusses convolutional autoencoders, recurrent autoencoders have similar bottleneck layers. The L2 distance metric is also applied to the LSTM-based RNNs tested in this research.

The attention matrices of a transformer can also be thought of as lower-dimensional representations of the input data, though not as direct a correlation. For the transformer

reconstructions, the same L2 distance is observed for each of the three attention layers of the basic transformer. This distance was not used for determining events in Part 2 of this research but should be further explored in Future Work.

Unlike the reconstruction errors above, the latent space distances are not explicitly using the model outputs. Instead, they are measuring the change of the internal representation. Matrices that are less similar correspond to a higher distance metric or greater amounts of change between two data windows. When entering an event window from steady-state, the latent space should be different and show increasing metric magnitudes. Again, a static threshold is calculated and any error sample that exceeds this threshold is considered part of an event.

Since autoencoders and transformers compress all of the data from the inputs together, there is no way to infer which parts of the lower-dimensional space correspond to which data trace. In each case, the L2 distance can only detect events on a system-wide basis.

3.6 Static Thresholding

As mentioned previously, events are detected when the reconstruction error or distance metrics exceed a static threshold. These thresholds are calculated from the training data then applied to the additional files at inference. Two methods were used for testing appropriate thresholds: mean plus standard deviation and a histogram method. The mean-based method calculated the mean and standard deviation of the training data at inference for the reconstruction error, first-difference, and various distance metrics (for individual

and system-wide traces when applicable). Then, the mean plus real number multiples of the standard deviation were set as the static thresholds when applied to the outputs of the test data. For the histogram-based method, a probability distribution of the metric outputs was calculated at an integer number of bin sizes based on the data's distribution. A bin number was selected based on model/metric performance and any error or distance exclusively greater than that bin is considered part of a detected event.

The mean-based method is shown as Equation 45 and the histogram-based method is shown as Equation 46.

$$\phi(x_t) \longmapsto \begin{cases} event, \ if \ \phi_{score}(x) > \mu_{train,c} + n_c \ \sigma_{train,c} \\ normal, \ otherwise \end{cases} \quad 45$$

where μ_{train} , σ_{train} are the mean and standard deviation calculated from the training data for each individual channel, c. The real number of standard deviations from the mean is n_c.

$$\phi(x_t) \longmapsto \begin{cases} event, \ if \ \phi_{score}(x) > binvalue(b_c, n_{bins}) \\ normal, \ otherwise \end{cases} \quad 46$$

where $binvalue$ is the value of the histogram bin in n_{bins} number of bins.

In the two cases, n_c and b_c are selected by the researcher based on the observed behavior of the error and/or distance traces.

Dynamic thresholding or more sophisticated methods to detect events from the error/distance space is beyond the scope of this research and should be explored in Future Work.

3.6.1 Determination Blur

In certain applications and domains, users may prefer or require an autonomous system to be more or less precise. Here, the choice is made to err on the side of wider events, including more time samples in a detection. The wider event is more likely to include a real event and still capture narrow events. Both methods can impact the objective metrics used to compare models. Additionally, an offset of a few time steps could be a discriminator between an actual event being detected or not when compared to a "truth" dataset (e.g. expert annotations). To allow for this necessary flexibility, an integer number of time steps before and after an error/distance space sample which exceeds the threshold will be included as part of the event. This is called determination blur in this paper and the parameter is set by the researcher before a model is presented to the participant for Part 2.

3.6.2 System-Wide Annotation Trace

Autoencoder- and transformer-based reconstructions create a lower-dimensional representation of the data. This compression has the potential to change how the interrelated behaviors are reconstructed between the multivariate data traces. For example, the trained model could learn a strong correlation between a bit-level monitor (e.g. transmission state) and a change in a related continuous analog signal (e.g. engine RPM). A POV is expected to spend most of its operational time in one transmission state (i.e. Drive), so certain behaviors could be viewed as normal in one transmission state and abnormal in another. These are contextual and collective anomalies. Similarly, specific changes in the distance measurements of the latent space cannot be directly associated with a single channel. This means that there is value in considering an aggregate event indicator

trace, which tracks if any trace indicates an event, with an additional system-wide evaluation metric.

Working with the expert annotations, a single system-wide event indicator trace can be constructed to compare with the system-level distance metric detections. In this trace, a time step is marked as part of an event if any of the individual traces flagged a detection at that time sample. Essentially, this flattens all annotations into a single $1 \times t$ trace where t is the total sample length of the data file.

3.7 Combined Expert Annotations

Whenever there are multiple individuals annotating a dataset, discrepancies are expected. Between annotators, major differences in what constitutes an "event," and minor differences in the particular sample which denotes the start or stop of a particular behavior, will manifest in how those annotators add markers in the toolset. This complicates objective comparisons between different time series reconstruction and event extraction options. A method to combine disparate annotations is required to create a "wisdom of the crowds" or ensemble of experts approach for a common baseline for comparison.

For this particular research and dataset, each expert's annotation file created in Part 1 (a time series of 1s and 0s) are added together for each individual collection file. If the summed value is greater than a threshold, i.e. a certain number of analysts deemed a particular sample is part of an event, then that time stamp is flagged as part of an event in the combined expert annotations file. Figure 15 shows an example.

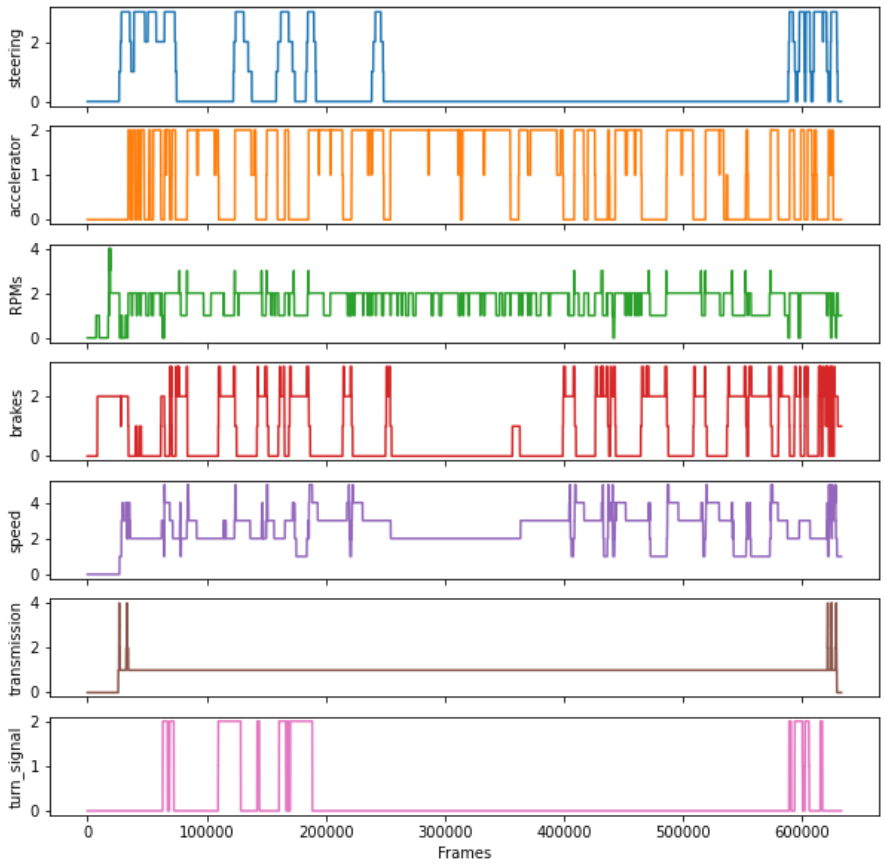


Figure 15: File 13 Sum of Annotation Traces.

This file will then be used for model evaluation metrics.

3.8 Evaluation Metrics Application

Where Chapter 2 defined evaluation metrics in the general case, this chapter specifically defines them for application in this thesis and CPS domain application.

The fundamental principle of this method is to autonomously determine which data points are a part of an event. When observing the time series, individual and/or the system-wide traces can be viewed as having a parallel boolean series of 1s and 0s where each

sample is or is not part of an event, respectively. In the case of the expert annotation files, these parallel boolean traces start with 0s until it reaches an "open" event marker then logs 1s until it finds the "close" marker. For the autonomous event extraction method, any sample which exceeds the reconstruction error or distance metric threshold is flagged as a 1 (an event) and all other samples are a 0.

These "event indicator traces" will be used during both quantitative and qualitative analysis and further discussed below.

$$E^{T\times C} = \begin{cases} 1, & if\ E_{t,c}\ is\ part\ of\ an\ event \\ 0, & otherwise \end{cases} \forall\ t \in T\ \forall c \in C \qquad 47$$

3.8.1 Comparing Event Indicator Traces - Boolean Algebra

Once the expert annotations or detected events are translated into their parallel events trace, it is critical to objectively compare two event indicator traces whether it is between two expert annotators (inter-coder agreement) or between the autonomous event extraction versus an expert's corrections (post-extraction correction). By treating one of the traces as "truth," i.e. all of the boolean 1s and 0s are correct, and comparing that event indicator trace against a test series, one can calculate a series of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) using boolean algebra (also called bitwise math or mathematical logic).

As mentioned previously, the basic operators are AND, OR, NOT, and XOR (exclusive OR) are used (Table 1).

Equations 48-51 combine these basic operators to generate the number of agreements (TP and TN) and disagreements (FP and FN) between two event indicator traces, $\boldsymbol{X}$ the test data and $\boldsymbol{y}$ the matrix of target/truth data for each channel, c.

$$TP_c = \sum_{t=0}^{T} AND\left(X_{t,c}, y_{t,c}\right) \quad 48$$

$$TN_c = \sum_{t=0}^{T} NOT\ \left(OR\left(X_{t,c}, y_{t,c}\right)\right) \quad 49$$

$$FN_c = \sum_{t=0}^{T} AND\ \left(y_{t,c}\ , XOR\left(X_{t,c}, y_{t,c}\right)\right) \quad 50$$

$$FP_c = \sum_{t=0}^{T} AND\ \left(X_{t,c}\ , XOR\left(X_{t,c}, y_{t,c}\right)\right) \quad 51$$

Lastly, the true positivity rate (TPR) and false positivity rate (FPR) are calculated by dividing the agreements and disagreements by the total number of positives and negatives.

3.8.1 F1-Score

F1-score balances precision with recall and is used as a metric of the overall agreement of two event indicator traces. It utilizes the true positive, true negative, false positive, and false negative metric established from Equations 48-51.

$$F_1 = \frac{2TP}{2TP+FP+FN} \quad 52$$

3.8.2 Cohen's Kappa

Cohen's kappa measures the agreement between two raters who classify N items into C mutually exclusive classes. This is another metric which balances the number of correct annotations against the total number of possible. Having only the choice of event or non-

event, this is a binary classification and can also be calculated with the Equation 48-51 metrics.

$$kappa = \frac{2\times(TP\times TN - FN\times FP)}{(TP+FP)\times(FP+TN)+(TP+FN)\times(FN+TN)} \quad 53$$

In this case, a value close to 1 means that coders agree more often, -1 means that the coders are more likely to disagree, and 0 means there is no correlation and the annotations are closer to chance.

3.8.3 Annotation Error

The annotation error is used to compare the total number of events detected and/or annotated. In the events trace, the annotation error is calculated by comparing the discrete number of 1s strings in each signal, combined signal, and/or distance metric.

$$AnnotationError_c = \left| K_{X,c} - K_{y,c} \right|$$

$$K_{trace,c} = \sum_{t=0}^{T} \begin{cases} 1, & if\ |E_{t,c} - E_{t-1,c}| = 1 \\ 0, & otherwise \end{cases} \quad \forall c \in C\ and\ trace \in \{X, y\} \quad 54$$

3.8.4 Hausdorff Distance

The Hausdorff distance is a measure of the longest string of event detections where two annotators or a validator and the system disagree. Essentially, the Huasdorff distance captures the worst annotation distance (spaced temporally) on an agreed upon event. The calculation of the Hausdorff distance executes a longest sub-sequence of 1's on an XOR of the two annotations.

3.8.5 Receiver-Operating Characteristic (ROC) Curve

Event/not event is a simple, binary classification task and the ROC curve will be used to show, graphically, the overall diagnostic ability of a selected detection method. The error- and distance-space thresholds, the discriminators, are enumerated together based on discrete fractions of the individual trace's standard deviations (calculated from each trace and the system-wide signal in the training data). The TPR and FPR can then be calculated at each step when the algorithm inference output is compared against the combined expert annotations (the "truth" data) for a given file.

Once the ROC is created, the area under the curve (AUC) is calculated as the objective metric to compare methods. The AUC was be used primarily to choose the detection method that will be presented to study participants in Part 2 of this research.

3.9 Summary

The event detection automation system represents one of the more important contributions to the target CPS domain. The system pre-processes raw CPS data (CAN data in this case), passes it through a lower dimensional latent space (either a CNN autoencoder, LSTM-based autoencoder, or transformer), reconstructs the data, and then takes reconstruction error and latent space distance metrics at each timestamp. If the metric or its first-difference exceed a calculated static threshold, an event is flagged. Boolean operations on these event indicator traces will be evaluated using: annotation error (a measure of the total agreement in numbers of events); Hausdorff distance (a measure of

the longest series of timestamps in disagreement); and F1-score (a measure which balances precision and recall for the events).

IV. Case Study

Across many applications of event detection, it is exceedingly difficult to objectively characterize an event without significant *a priori* information about the domain. Different human annotators and/or validators could also have varying opinions on what constitutes an event, which traces should be considered when making the characterization, and/or be biased/constrained by varying levels of trust, required certainty, or mission requirements. Therefore, any autonomous system is unlikely to replace a human entirely. It is merely a tool to improve accuracy and accelerate analysis.

The *participants* in the case study are CPS analysis domain experts.

There are two components of this mixed methods research. The first is a quantitative study that evaluates accuracy of objective metrics describing how well unsupervised machine learning locates/extracts events. Accuracy is measured by how well EDAS identifications align with expertly annotated events. The second portion of this study is a qualitative survey of potential users to determine time savings and perceived utility of the process.

The research design engages domain experts in two parts consisting of four steps. Part 1 includes a pre-survey as Step 1 and asks the experts to annotate a subset of the overall collected data for Step 2. This creates a baseline of both qualitative and quantitative test data to characterize the time required to perform analysis (using the legacy method), inter-coder agreement, and effectiveness of the unsupervised event extraction methods. The survey of Part 1 further establishes a window into participants' initial perceptions of where in their process autonomous event extraction methods could save them the most time. Part 2 utilizes a pre-selected model architecture for the EDAS and autonomously generates

event files. The same domain experts were asked to validate/correct the extracted event markers (Step 3) and fill out a post-survey (Step 4). The second part provides quantitative data on the effectiveness of the autonomous event extraction–plus cross-validation with the initial Part 1 annotations–and qualitative insight into how users perceive the utility of the EDAS.

4.1 Case Study Overview

The qualitative case study evaluates machine-learning techniques against existing analytic processes and hopes to accelerate the existing, labor-intensive event detection methods. The study consists of two parts and involves subject matter experts using their domain knowledge and experience to interpret, analyze, annotate, and correct the presented data files. Their expert opinions are well-informed, but subjective. With pre- and post-surveys, participant demographics are captured plus their initial and *post facto* impressions of the impact of automation and the event detection algorithm on their workflow.

Figure 16 shows the general case study design.

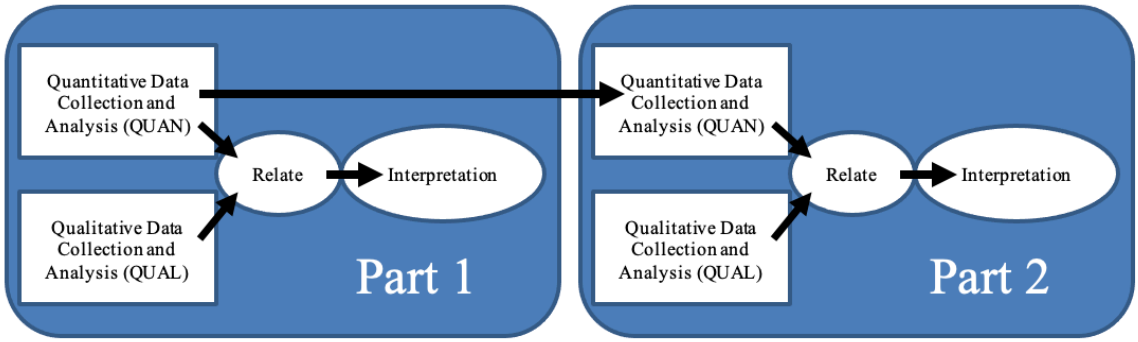


Figure 16: Mixed Methods Case Study Design.

4.1.1 Case Study Data Plan

This research uses a mixed method approach including a qualitative and quantitative look at the data. Following the best practice of using segregated train and test data, each participant was assigned a different subset of the collected files for Part 1. Following their initial annotations, that data was used to train the selected deep-learning model and perform parameter tuning. The remaining files were then provided to the participant for Part 2 for validation and correction. Table 6 shows how these files were assigned.

Table 6: Case Study Data Plan.

	Part 1						Part 2					
Participant 01	File 01	File 02	File 03	File 04	File 05	File 06	File 07	File 09	File 10	File 11	File 12	File 13
Participant 02	File 01	File 03	File 04	File 05	File 06	File 09	File 07	File 10	File 11	File 12	File 13	File 02
Participant 03	File 01	File 04	File 05	File 06	File 09	File 10	File 07	File 11	File 12	File 13	File 02	File 03
Participant 04	File 01	File 05	File 06	File 09	File 10	File 11	File 07	File 12	File 13	File 02	File 03	File 04
Participant 05	File 01	File 06	File 09	File 10	File 11	File 12	File 07	File 13	File 02	File 03	File 04	File 05
Participant 06	File 01	File 09	File 10	File 11	File 12	File 13	File 07	File 02	File 03	File 04	File 05	File 06
Participant 07	File 01	File 10	File 11	File 12	File 13	File 02	File 07	File 03	File 04	File 05	File 06	File 09
Participant 08	File 01	File 11	File 12	File 13	File 02	File 03	File 07	File 04	File 05	File 06	File 09	File 10
Participant 09	File 01	File 12	File 13	File 02	File 03	File 04	File 07	File 05	File 06	File 09	File 10	File 11
Participant 10	File 01	File 13	File 02	File 03	File 04	File 05	File 07	File 06	File 09	File 10	File 11	File 12

All participants were assigned File 01 for Part 1 to evaluate at inter-coder agreement across all of the participants. File 07 was assigned to everyone during Part 2 to similarly create a control file and facilitate comparisons across every expert. File 08 was not used because the collection included activity that was not typical "ambient" driving, i.e. the POV operator was waiting in a drive-thru line. Analysis of this behavior was determined to be out-of-scope for this study (but may be explored in subsequent research). By assigning the remaining files in a rotating fashion, the experts have a reduced time burden, there is

segregated training and testing data, and there is Part 1/Part 2 cross-validation between legacy method annotations versus autonomous event extractions.

4.1.2 Inter-Coder Agreement

Inter-coder agreement is a quantitative evaluation for the qualitative contributions of the participants. Human-labeling is inherently noisy and each annotator will disagree with each other and with the event detection automation system. The inter-coder agreement is measured using F1-score and Cohen's kappa calculated from the binary event indicator traces.

4.1.3 Survey Evaluation

A qualitative assessment of Likert-like scales and short answers was conducted to search for trends between participant demographic, their familiarity with AI/ML methods, and overall trust in the system. The final evaluation of the overall utility of this EDAS is judged on the concluding survey to capture participants' perceptions of the interaction with the autonomous event detection method.

4.2 Part 1 - Legacy Methods and Baseline Inter-Coder Agreement

The first part of the study asked participants to perform their normal analysis process on the case study dataset, in their natural setting and using their typical tool sets. The pre-survey is primarily intended to capture their initial thoughts on AI/ML techniques in general and specifically applied to their analytic domain. Their expert annotations serve as a baseline for inter-coder agreement and are used to construct the combined expert annotations for Part 2.

4.2.1 Pre-Survey

As described in the Experimental Design, Part 1 begins with a pre-survey, the first engagement with the experts. This survey is broken into three parts:

1. **Respondent information** (Likert-like scale): which attempts to capture demographic information and initial familiarity with, and trust of, machine-learning applications.
2. **Current analytic methods** (Likert-like scale): which asks questions about the participants' current event extraction methods and establishes a baseline for time and effort.
3. **Initial perceptions** (Likert-like scale and short answer): which tries to characterize the experts' perceptions of how difficult the event extraction problem is and how useful a working, autonomous system might be.

Appendix A.2 contains the Part 1 instrument and the IRB approval.

The survey questions attempt to capture a snapshot of a participant's domain experience as well as their prior experience with AI/ML, domain expertise, and perception of the autonomous event detection problem. The later questions are used to determine if their prior experiences influence their opinions and interactions with the EDAS.

4.2.2 Expert Annotations Study

Both Part 1 and Part 2 of this research leverage participants to annotate training data then correct the outputs of the autonomous event extraction method. Using established domain analysis tools and techniques, these experts annotate frame-based markers which indicate the start and end of events in each individual parameter signal. For example, an

engine turn-on event would be initiated by pressing the brake (marker 1 in channel X), followed by a stark increase from zero in the engine RPMs (marker 1 in channel Y), then the RPMs would settle to idle (marker 2 in channel Y), and finally the brake pressure would decrease (marker 2 in channel X). These markers are treated as binary switches which consider any sample between start and end markers as part of an event, marked as a 1, and not part of an event otherwise, marked as a 0.

4.2.2.1 Baseline Inter-Coder Agreement

During Part 1, a baseline is established by comparing the expert annotations for each individual data file. In this research application, File 01 is annotated by every participant and establishes an overall agreement metric between the participants. The additional files are used to show annotation stability across various coders for each file individually. In each case, the hand-labels of the coders will be evaluated three times–once with each of the two annotators operating as the "truth" label (F1-score) and once with the Cohen's kappa metric.

4.2.2.2 Combine Expert Truth

Due to the data plan, all of the files have five annotations except File 01 (which every participant annotates) and File 07 (which every participant corrects). The combined expert annotations for these files are also calculated following the collection of these initial labels. This served as an objective baseline for measuring the accuracy of EDAS.

4.3 Parameter Sweep and Model Selection

One of the primary goals of this research is to deliver an accurate, reliable, and trustworthy set of autonomously detected events to a human participant/validator. Part of this requirement is to perform a parameter sweep for the window size, data stride, model architecture, and event detection metric that performs best in this domain.

The parameter sweep uses window sizes of 64, 128, and 256 samples with data strides of 12.5%, 25%, 50%, and 100% of the window size. For example, there were data sets of 64x8, 64x16, 64x32, and 64x64. Three model architectures were tested for both the CNN and LSTM-based RNN autoencoders: 32x16x32, 64x32x64, and 64x32x16x32x64. Each number corresponds to a number of convolutional or LSTM layers, respectively, and the center number is the size of the latent space or "bottleneck." For the transformers, Time2Vec vector embedding sizes of 16, 32, and 64 with the direct feed-forward layers of 512 and 1024 nodes. There were four transformer models considered 16x512, 32x1024, 64x512, and 64x1024.

The AUC metric was used for comparison between each input size (as expressed by the window and stride) and model architecture. This represented the overall discrimination capacity for each set of hyperparameters and was the basis for the model selection–the highest AUC was decided to be optimal. Once chosen, the parameters were used when the experts ran Part 2: executed the selected model, created a set of autonomous event detection output files, and human corrected the output.

This list is not intended to be exhaustive, but only to recognize general trends between input, model, and event detection performance. Additional hyperparameter searches may yield better results, but this further exploration is beyond scope.

4.4 Part 2 - Unsupervised Event Detection

Part 2 captured participants' interactions with the autonomous event detection method. Ultimately, the goal was to: a) deliver utility, through high F1-score and low annotation error; b) accelerate processes, through direct time savings; and c) be trusted by participants (as evidenced by high inter-coder agreement between the event detection automation system and participant). Case study participants ran the autonomous system, then corrected the EDAS detections in their natural settings and existing analysis tools. A post-survey captured their final perceptions of the method and the factors which influenced their trust in the system.

4.4.1 Post-Detection Corrections Study

Part 2 required participants to run the EDAS method, train the selected model using their list of Part 1 files as their training set (the annotations were not used). This resulted in ten individually trained EDAS models which output autonomous detections for each participant's test set. The participants then corrected the event detections output by adding, subtracting, or moving detection markers. Both the original system output and the corrected events time series were saved for analysis. For each file and participant model pairing, inter-coder agreements were calculated between:

- the autonomous system output and its combined expert annotations
- the autonomous system output and the participant's correction
- the participant's correction with the combined expert annotations

Each of the agreement metrics above yielded a different view on how the participants interacted with the event detection automation system and compared with each other. Similar to the baseline above, File 07 was corrected by every coder. This also gave an agreement and stability metric across every expert when the same data is output from the same model architecture but trained on different data.

4.4.2 Post-Survey

After Part 2, and the last step in this research, the experts filled out the post-survey. This instrument used two parts:

1. **Method interactions** (Likert-like scale and short answer): which characterizes the ease of the program use and approximate time savings.
2. **Method perceptions** (Likert-like scale and short answer): which captures the participant's qualitative views of the method and what factors influences their trust in the system.

The Part 2 instrument can be found in Appendix A.3 with the IRB information.

These questions were intended to record their perceptions of system accuracy and usefulness of the method. The open-ended trust questions captured potential pre-method training to better understand the system outputs and areas of improvement for follow-on research or Future Work.

4.5 Full Event Detection Automation System Evaluation

The final evaluation of the overall system and perceived utility for domain experts involves a comprehensive, qualitative discussion of the quantitative analysis of each model

against combined expert annotations (e.g. F1-score, AUC, and training time), legacy extraction method timeframes, time required at inference, inter-coder agreements, and survey response data. The number of files and experts will be too small for broad generalizations, even for this narrow domain. However, this research could be used for a baseline when developing more sophisticated tools or techniques which use unsupervised automation.

4.6 Summary

The case study is an important method for determining whether EDAS will deliver utility to participants. Part 1 studied the legacy, manual task of event annotation in the CPS data and participants' initial perceptions of AI/ML as it applies to the autonomous detection task. Part 2 studied the participants' interactions with EDAS through a validation and correction task then collects survey responses about their interactions. Quantitative inter-coder agreement analysis will use F1-score, kappa, annotation error, and Hausdorff distance.

V. Analysis and Results

The research questions (RQ) are: 1) Can EDAS autonomously detect events? 2) Can EDAS provide accurate and reliable that develop trust? 3) Can EDAS save time, compute, and/or analytic burden? This chapter discusses the quantitative and qualitative results of Part 1 and Part 2 of the case study plus the parameter sweep results for event detection automation system (EDAS).

For this research, the selected model architecture was a CNN autoencoder with a 32x16x32, 64x16 window/stride, and using the trace-wise detections (since the system-wide measures underperformed in nearly every model). The qualitative research showed the case study participants were cautiously optimistic about the autonomous method but were not impressed with its current state. However, the inter-coder metrics showed low agreement across every file in both Part 1 and Part 2, calling into question the likelihood of finding any autonomous system which would satisfy all analysts.

This Chapter concludes with a general discussion on the unsupervised, autonomous event detection and the accompanying case study.

In the presented tables, the colors are used to distinguish a gradient spectrum of results that are considered a “positive” or “negative” result in relationship to the research questions as green or red, respectively.

5.1 Part 1 Results

The intent of Part 1 is to establish a baseline of both participant perceptions and baseline metrics of their inter-coder agreements. Generally, most participants believe that

event detection is a time-consuming, difficult problem and agree that automation would likely offer positive benefits in their analysis domain. However, there was not much consensus among the coders on the initial file annotations and there is not *a priori* knowledge of AI/ML methods, both would likely negatively impact the eventual acceptance of any autonomous solution. Further discussion is below.

5.1.1 Pre-Survey Results

Table 8 shows the responses provided by the ten participants. Short answer themes will be discussed as well.

5.1.1.1 Likert-like Scale Questions

Starting with the Likert-like scale questions, a few points are discussed here.

1. **Participants Consider Current Manual Methods Only Somewhat Difficult (RQ 1)**

According to Question 2.1 responses, most participants rated their current man-in-the-loop methods as 'minimally difficult' to 'difficult'. Additionally, 5 of the 10 participants stated that only 5% of a typical week is used doing event detection (Question 2.4). Though not rated particularly difficult, even an easy task performed several times can be cumbersome. Coupled with observation 3 below, it seems savings could be realized by offloading some of this task to autonomous computation. When considering research question 3, low perceived difficulty of the manual task may limit the perceived effort savings.

Table 7: Pre-Survey Results

		Participant Number									
	Question	01	02	03	04	05	06	07	08	09	10
1.1	How many years of experience do you have analyzing time series data?	2	2	5	2	2	2	1	1	4	2
1.2	How confident are you in your ability to analyze time series data?	4	3	5	4	3	4	2	1	5	3
1.3	How familiar are you with artificial intelligence, machine-learning, or deep-learning concepts/research?	3	2	3	3	3	3	2	3	3	2
1.4	How important do you believe autonomous event extraction could be for your application domain?	2	4	5	5	4	5	5	4	5	5
1.5	How confident are you that machine-learning methods can accelerate your analytic workflow for event extraction?	3	4	4	4	3	4	5	4	5	4
2.1	How difficult would you characterize man-in-the-loop event extraction?	4	3	3	5	3	2	3	4	2	3
2.2	2.2. How long do you estimate it takes to manually extract events from a typical formatted, but new data file for a system you are familiar with?	1	3	1	5	2	2	4	3	2	-
2.3	How long do you estimate it takes to manually extract events from a typical formatted, but new data file for a system you are unfamiliar with?	3	5	5	5	5	5	5	5	3	-
2.4	What percentage, during an average week, do you estimate you spend extracting events from formatted data?	1	3	1	5	3	1	1	1	2	-
2.5	How confident are you that your current process methods catches all relevant "events"?	4	5	2	4	4	4	2	1	4	-
2.6	How timely would you say your current methods are? i.e. Do your current methods output analysis deliverables (e.g. reports or models) at the speed required by customers?	4	4	2	5	2	3	1	4	5	-
3.1	How difficult would you characterize autonomous (computer-based) event extraction?	2	4	2	5	5	1	3	4	3	5
3.2	How much time savings do you believe future autonomous extraction methods could save during an average week when considering the required validation of automated outputs?	2	3	5	5	5	3	5	5	4	4
3.3	Describe your experiences with past efforts at automating event extraction automations.	SHORT ANSWER									
3.4	Describe your concerns with automated event extraction methods.	SHORT ANSWER									

2. Most Participants Were Relatively Inexperienced, Somewhat Familiar With AI/ML Applications (RQ 2)

The first observations involved the demographics of the participants. Most of the participants have less than 5 years of experience in time series analysis (Question 1.1).

Only two exceeded this threshold. However, most were at least minimally confident in their ability to analyze the data (Question 1.2) and confident their methods pulled out all relevant events (Question 2.5). All surveyed participants were either 'unfamiliar' or only 'somewhat familiar' with AI/ML concepts and research (Question 1.3).

3. Event Detections in Familiar Data is Quick but Much Slower in Unfamiliar Data (RQ 3)

In most cases, participants stated that their manual methods took between 5-30 minutes per file when annotating data that they are familiar with (Question 2.2). However, that number grew to over an hour for 7 of 10 when applied to data that is not familiar (Question 2.3). Accelerating the early identification of possible events in unfamiliar data could represent the most potential for injecting autonomous learning (research question 3).

4. Participants See Potential Value and Time Savings in Autonomy Aids (RQ 3)

The last observations are core to this research application. First, 9 of 10 participants believe autonomous event extraction could be 'important' or 'very important' (Question 1.4). Secondly, most participants believe autonomous methods could save up to, or greater than, 10% of time for their average week–even considering the time required to correct automated detections (Question 3.2). Both show that this is a valid research area which could directly impact the effectiveness of the population. The participant optimism shows the value of considering research question 3.

5.1.1.2 Short Answer Questions

The open-ended responses to Question 3.3 and Question 3.4 showed a few trends. Though skeptical about the expected performance, participants seemed cautiously optimistic about the potential of autonomous event detection. A few more key themes are discussed below.

1. **Concerned with Data Availability and Computing Resources (RQ 1)**

Machine learning applications are known for needing large amounts of data and require more computing resources than a typical domain task. Though the participants were less familiar with AI/ML concepts (Question 1.3), they expressed a concern that, in a data and compute constrained environment, there may not be sufficient resources to generate an effective method.

2. **Skeptical of System Performance in the Presence of Noise (RQ 2)**

Secondly, several participants specifically discussed the presence of noise in event detection problems. When discussing previous experiences with automation attempts (Question 3.3), some mentioned how basic tiered thresholding of the raw data is significantly impacted by noise. Even with a low bit error rate, this often leads to significant over-segmenting and/or event detection triggers–especially in continuous, analog signals. Considering noise will likely impact participant perceptions of EDAS (research question 2).

3. **Events Need to Be Well-Defined Across Targets (RQ 2)**

Mentioned several times throughout this thesis, event detection methods are often domain specific. There can be significant variability between targets. Participants expressed concern that the methods will be limited to a particular target and not necessarily applicable for multiple systems. Additionally in application domains beyond CAN analysis, there are many data formats, sampling rates, and event shapes that can all be observed in the same system.

4. **Unlikely to Fully Replace the Human (RQ 3)**

The last idea shared by many of the participants was that any automation cannot fully replace a human. This is a perception shared by the researcher. When decision-makers rely on automated extractions, false events could create confusion or bad outcomes due to wrong information. Especially for high-impact/high-consequence event activity, or interpreting the meaning of multiple events, human insight is necessary to correct and assure correct assessments. Additionally, one participant was concerned with how to monitor drift of the model over long periods of time. Both factors led to skepticism about whether automated methods will save them time.

5.1.1.3 Pre-Survey Discussion

The cautious optimism expressed by the survey participants shows that this is a valid research area which could fit within these participants' workflows and deliver time- and effort-saving utility. However, the participants expressed valid concerns.

First, the small data volumes of this test case and that are typically found in the participants' application domains indeed create problems for most machine learning methods. Deep autoencoders and transformers specifically used to help alleviate this issue. Narrow, shallow autoencoders can be designed to underfit even small datasets and are known to perform well on smaller numbers of samples. Data augmentation methods could be used, but may introduce additional concerns.

Secondly, participants were concerned that events would need to be well-defined across multiple diverse targets. It is unclear if the participants were referring to transfer learning, applying an identical or similar model to another target, or the need for a model to learn all the possible behaviors for a given target. Regardless, the unsupervised nature of this method is intended to independently learn the representative data models with minimal *a priori* domain knowledge or participant input. This method is also intended to be part of the end-to-end analytic workflow in Figure 1. It does not need to learn everything about the system before application because it will be continuously learning through the active learning process. The short training time demonstrated for these methods (discussed further below) lowers the cost for retraining in different domains.

Lastly, unsupervised event detection is not meant to replace human analysts but to accelerate their tedious annotation processes. Coupled with fundamental misunderstandings about the nature of unsupervised methods, this shows education and increased AI/ML literacy could be a key determinant for the eventual adoption of any autonomous system in this analytic domain.

5.1.2 Baseline Annotation Results

The expert annotation study asks the participants to use legacy methods to manually perform the event extraction task. As mentioned in Chapter 4, the annotators add event annotations based on their domain expertise and in their natural analytic setting. These markers bracket any events identified in each individual trace and their aggregate make up the system-wide, participant-derived events. Sample time series representations of these annotations are shown in Appendix C.

Through either analysis decisions or process errors, several participants did not annotate every trace within a given file. Others failed to "close" an event which caused all subsequent labels to be flipped. Though not excessively present and assuming these are incorrect markings, this impacted the overall accuracy of the inter-coder agreements. However, the inter-coder metrics were quite low despite these issues, calling into question the stability and reliability of human annotations as a baseline metric.

5.1.2.1 File 01 - All Participants Inter-coder Annotation Agreement

To establish a baseline for inter-coder agreement, all participants were tasked with annotating File 01. The first two metrics are the F1-score and Cohen's kappa shown in Table 8 and Table 9.

Table 8: Inter-coder Annotation Agreement, F1-score.

F1		Truth Labels										
File 01		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	1.0000	0.2914	0.2240	0.2777	0.3557	0.1796	0.1261	0.1220	0.0551	0.1688	0.2000
	02	0.2914	1.0000	0.5159	0.5164	0.2961	0.1831	0.1134	0.0547	0.0874	0.1438	0.2447
	03	0.2240	0.5159	1.0000	0.8200	0.5855	0.1832	0.0684	0.0628	0.1667	0.1057	0.3036
	04	0.2777	0.5164	0.8200	1.0000	0.6683	0.1970	0.0971	0.0596	0.1657	0.1093	0.3235
	05	0.3557	0.2961	0.5855	0.6683	1.0000	0.1067	0.0917	0.0769	0.0935	0.1089	0.2648
	06	0.1796	0.1831	0.1832	0.1970	0.1067	1.0000	0.0694	0.0791	0.0838	0.1197	0.1335
	07	0.1261	0.1134	0.0684	0.0971	0.0917	0.0694	1.0000	0.4455	0.0429	0.1890	0.1382
	08	0.1220	0.0547	0.0628	0.0596	0.0769	0.0791	0.4455	1.0000	0.1676	0.5634	0.1813
	09	0.0551	0.0874	0.1667	0.1657	0.0935	0.0838	0.0429	0.1676	1.0000	0.2051	0.1186
	10	0.1688	0.1438	0.1057	0.1093	0.1089	0.1197	0.1890	0.5634	0.2051	1.0000	0.1904
Mean		0.2000	0.2447	0.3036	0.3235	0.2648	0.1335	0.1382	0.1813	0.1186	0.1904	0.2099

Table 9: Inter-coder Annotation Agreement, kappa.

Kappa		Truth Labels										
File 01		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	1.0000	0.2659	0.1990	0.2546	0.3378	0.1331	0.0794	0.0681	0.0015	0.1183	0.1620
	02	0.2659	1.0000	0.4987	0.4993	0.2750	0.1277	0.0574	-0.0159	0.0254	0.0807	0.2016
	03	0.1990	0.4987	1.0000	0.8143	0.5742	0.1378	0.0196	0.0065	0.1204	0.0526	0.2692
	04	0.2546	0.4993	0.8143	1.0000	0.6592	0.1530	0.0503	0.0039	0.1199	0.0571	0.2902
	05	0.3378	0.2750	0.5742	0.6592	1.0000	0.0681	0.0541	0.0345	0.0542	0.0683	0.2362
	06	0.1331	0.1277	0.1378	0.1530	0.0681	1.0000	-0.0948	-0.2051	-0.1164	-0.1372	0.0074
	07	0.0794	0.0574	0.0196	0.0503	0.0541	-0.0948	1.0000	0.3169	-0.1265	0.0087	0.0406
	08	0.0681	-0.0159	0.0065	0.0039	0.0345	-0.2051	0.3169	1.0000	-0.0907	0.3528	0.0523
	09	0.0015	0.0254	0.1204	0.1199	0.0542	-0.1164	-0.1265	-0.0907	1.0000	-0.0280	-0.0045
	10	0.1183	0.0807	0.0526	0.0571	0.0683	-0.1372	0.0087	0.3528	-0.0280	1.0000	0.0637
Mean		0.1620	0.2016	0.2692	0.2902	0.2362	0.0074	0.0406	0.0523	-0.0045	0.0637	0.1319

In both tables, the participants are compared against each other–once where the first participant is the notional truth annotator and once where it is the second. These metrics are both measuring the similarities between the boolean event indicator traces. Ignoring the diagonal (where a participant is measured against themself), the F1-score varies between 0.0429 and 0.8200 (mean of 0.2099) and the kappa varies between -0.2051 and 0.8143 (mean of 0.1319). Analyst 03 and Analyst 04 have strong correlation (F1 0.8200 and kappa

0.8143). Generally, researchers are looking for an 80% agreement [58] or a kappa between 0.40-0.60 [60], so the 0.1319 is indicative of very low inter-coder agreement.

The next measure, annotation error shown in Table 10, similarly shows a large number of disagreements in the total number of annotated events (especially Analyst 07 and Analyst 08). The average difference in the number of total events is 121, which is high, but six of ten participants were closer to 30 overall events.

Table 10: Inter-coder Annotation Agreement, Annotation Error.

Error		Truth Labels										
File 01		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	0	37	25	17	26	20	225	272	33	73	81
	02	37	0	12	22	31	31	236	277	36	62	83
	03	25	12	0	10	19	33	240	275	32	54	78
	04	17	22	10	0	9	27	234	285	24	64	77
	05	26	31	19	9	0	34	241	294	33	73	84
	06	20	31	33	27	34	0	207	286	37	81	84
	07	225	236	240	234	241	207	0	403	230	196	246
	08	272	277	275	285	294	286	403	0	287	231	290
	09	33	36	32	24	33	37	230	287	0	68	87
	10	73	62	54	64	73	81	196	231	68	0	100
Mean		81	83	78	77	84	84	246	290	87	100	121

The final baseline metric for File 01 is the Hausdorff distance or the longest length of disagreement between overlapping events. This is shown in Table 11.

Table 11: Trace-Wise Hausdorff Distances for Part 1 Annotations.

Hausdorff	Truth Labels										
Steering	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	5631	9629	4258	3020	13488	6175	124954	0	8538	19521
2	5631	0	4609	4839	5668	13488	3612	124892	0	3183	18436
3	9629	6023	0	6549	9666	13488	2942	124954	0	6396	19961
4	9659	8832	4466	0	9714	13488	1541	124654	0	5658	19779
5	2491	6524	9666	4418	0	13488	6212	124954	0	8575	19592
6	9659	6501	2165	5154	9714	0	2942	124954	0	3594	18298
7	6175	3612	2942	1389	6212	13488	0	124954	0	2942	17968
8	14908	124892	12645	124654	10875	13174	10225	0	0	124909	48476
9	9659	9335	6467	8537	9714	13488	5679	124954	0	9225	21895
10	8538	3183	2909	4571	8575	13488	2942	124909	0	0	18791
Mean	8483	19393	6166	18263	8129	13453	4697	124909	0	19224	24746

Accelerator	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	14867	2491	0	0	0	0	22833	0	152036	21359
2	0	0	13938	0	0	0	0	22833	0	136813	19287
3	0	14832	0	0	0	0	0	22833	0	150751	20935
4	0	14867	2491	0	0	0	0	22833	0	152036	21359
5	0	14867	2491	0	0	0	0	22833	0	152036	21359
6	0	14867	2491	0	0	0	0	22833	0	152036	21359
7	0	14867	2491	0	0	0	0	22833	0	152036	21359
8	0	13127	2527	0	0	0	0	0	0	18403	3784
9	0	14867	2491	0	0	0	0	22833	0	152036	21359
10	0	136813	150751	0	0	0	0	18403	0	0	33996
Mean	0	28219	20240	0	0	0	0	22341	0	135354	22906

Brakes	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	1114	540	0	0	7575	37462	158293	0	21438	25158
2	0	0	627	0	0	6623	37462	158293	0	21438	24938
3	0	627	0	0	0	7250	37462	158293	0	21438	25008
4	0	1114	540	0	0	7575	37462	158293	0	21438	25158
5	0	1114	540	0	0	7575	37462	158293	0	21438	25158
6	0	6623	7250	0	0	0	37462	158293	0	21438	25674
7	0	19481	315	0	0	12858	0	16698	0	25840	8355
8	0	619	151	0	0	12970	16698	0	0	158021	20940
9	0	1114	540	0	0	7575	37462	158293	0	21438	25158
10	0	6525	7152	0	0	13291	37462	158021	0	0	24717
Mean	0	4259	1962	0	0	9255	35155	142530	0	37103	25585

Speed	Truth Labels										
	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	13477	13477	13477	0	14195	9073	31941	47134	150642	32602
2	13477	0	376	1096	0	14891	15053	31941	56835	158934	32511
3	13477	376	0	1095	0	14891	15053	31941	56648	158760	32471
4	13477	1272	1226	0	0	14891	14619	31941	56364	158762	32506
5	9145	15905	15924	15031	0	16242	3975	31941	56835	159960	36106
6	14195	1272	1226	9800	0	0	15053	31941	47345	148385	29913
7	9073	15053	15053	14619	0	15053	0	31941	56207	156001	34778
8	19549	15740	15818	9499	0	15806	20101	0	11743	45092	17039
9	47134	20698	56648	56364	0	47345	56207	31941	0	158459	52755
10	150642	158934	158760	158762	0	148385	156001	45092	158459	0	126115
Mean	32241	26970	30945	31083	0	33522	33904	33402	60841	143888	47422

RPMs	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	0	0	0	0	0	1897	21791	86896	21894	14720
2	0	0	0	0	0	0	1897	21791	86896	21894	14720
3	0	0	0	0	0	0	1897	21791	86896	21894	14720
4	0	0	0	0	0	0	1897	21791	86896	21894	14720
5	0	0	0	0	0	0	1897	21791	86896	21894	14720
6	0	0	0	0	0	0	1897	21791	86896	21894	14720
7	0	0	0	0	0	0	0	21791	79381	21894	13674
8	0	0	0	0	0	0	8463	0	65770	6135	8930
9	0	0	0	0	0	0	66713	65770	0	65236	21969
10	0	0	0	0	0	0	2959	6135	65236	0	8259
Mean	0	0	0	0	0	0	9946	24938	81307	24959	15683

Transmission	Truth Labels									
	1	2	3	4	5	6	7	8	9	10
Test Labels 1	0	931	14644	13423	13330	678450	775	1075	13679	3238
2	931	0	14187	14187	14166	678392	223	92	14187	1771
3	14644	14187	0	1221	8200	677200	13900	14059	8200	11489
4	13423	14187	1221	0	8455	678402	13900	14059	8462	11489
5	13330	14166	8200	93	0	678450	13900	14027	940	11489
6	13391	678392	677200	678402	8455	0	678131	678404	677323	675527
7	775	223	13900	13900	13900	678131	0	234	13900	1617
8	1075	97	14059	14059	14027	678404	300	0	14059	1770
9	13679	14187	8200	8462	940	677323	13900	14059	0	11489
10	462	1771	11489	11489	11489	675527	1617	1770	11489	0
Mean	7968	82016	84789	83915	10329	677809	81850	81975	84693	81098

Turn Signal	1	2	3	4	5	6	7	8	9	10
Test Labels 1	0	0	14106	14792	14792	0	0	14792	261883	14792
2	1491	0	14468	16217	15712	0	0	15452	278069	15705
3	14106	0	0	1405	1023	0	0	1011	263739	1041
4	14792	0	1405	0	978	0	0	987	261730	943
5	14792	0	1023	978	0	0	0	133	262247	45
6	1491	0	14468	16217	15712	0	0	15452	278069	15705
7	1491	0	14468	16217	15712	0	0	15452	278069	15705
8	14792	0	1011	987	133	0	0	0	262412	171
9	261883	0	263739	261730	262247	0	0	262412	0	262197
10	14792	0	1041	943	45	0	0	171	262197	0
Mean	37737	0	36192	36610	36262	0	0	36207	267602	36256

5.1.2.2 Inter-coder Annotation Agreement Across Other Files

Using the Case Study Data Plan shown in Table 6, Table 12 shows the averages of the F1-score, Cohen's kappa, and annotation error .

Table 12: Summary of Part 1 Inter-coder Annotation Agreement.

	Baseline	File Number									
	01	02	03	04	05	06	09	10	11	12	13	Mean
F1-score	0.2099	0.3002	0.0803	0.2118	0.3444	0.3097	0.3643	0.2973	0.2148	0.2899	0.2095	0.2622
kappa	0.0637	0.1396	0.0413	0.1417	0.3116	0.2909	0.3374	0.2072	0.0887	0.1138	0.0672	0.1739
Error	121	131	77	19	30	18	15	48	69	98	112	62

These values show similarly low agreement as the baseline across every metric.

5.1.2.3 Annotation Time Analysis

Part of the stated goals of this research is to accelerate the event detection process. This section sets a baseline for the time required to perform the manual extraction process in the participants' natural setting. Research question 3 was that EDAS would reduce time, so annotation time analysis is the baseline to compare with.

Table 13 depicts the time required by participants to perform the initial annotation task per minute of the collected data file. Using the User Study Data Plan in Table 6, participants only interact with a subset of the data during each Part. When applicable, this is annotated in subsequent tables with blank, greyed entries.

Table 13: Annotation Time for Part 1.

		File Number													
File Time (mins)		01	02	03	04	05	06	07	08	09	10	11	12	13	
		26.5	27.4	20.3	3.77	20.5	18.5	19.8	27.6	14.9	14.6	12.8	22.3	23	
Participant Number	01	0.756	0.365	0.247	0.796	0.244	0.379			-	-	-	-	-	0.465
	02	2.042	-	1.037	3.979	0.684	0.759			0.806	-	-	-	-	1.551
	03	0.756	-	-	5.305	0.830	1.138			1.075	1.441	-	-	-	1.758
	04	0.567	-	-	-	0.391	0.379			0.403	0.480	0.312	-	-	0.422
	05	0.302	-	-	-	-	0.271			0.202	0.343	0.234	0.269	-	0.270
	06	0.983	-	-	-	-	-			0.605	0.686	1.169	0.627	0.610	0.780
	07	1.323	0.329	-	-	-	-			-	1.441	0.857	0.403	0.479	0.805
	08	1.134	1.278	1.235	-	-	-			-	-	3.117	1.343	1.525	1.605
	09	0.378	0.183	0.247	1.326	-	-			-	-	-	0.358	0.436	0.488
	10	N/A	1.753	2.519	4.775	1.660	-			-	-	-	-	2.266	2.594
Mean		0.916	0.782	1.057	3.236	0.762	0.585	N/A	N/A	0.618	0.879	1.138	0.600	1.063	1.058
StdDev		0.539	0.694	0.933	2.049	0.553	0.360	N/A	N/A	0.341	0.528	1.172	0.436	0.806	0.765

These values range between 0.183 and 5.305 minutes to annotate every minute of data in the file (mean 1.0577). The highest ratios were with the shortest file (File 04), but longer durations do occur in other files. On average, it takes slightly longer, about 6%, to manually extract events per minute of collection. Considering most participants likely analyzed files in numerical order, the data does not show an acceleration due to familiarity with the method, data, or event behavior.

5.1.2.4 Combined Annotation Truth Metrics

The combined annotation truth generates a baseline events trace to objectively compare the autonomous detection method. This should be a better overall representation of the true annotated events, a “wisdom of the crowds” approach. In this application, if three coders agree that a given data point is an event, then the combined expert annotations

labels that sample to be part of an event. Figure 17 shows an example file with the number of agreements on the secondary y-axis.

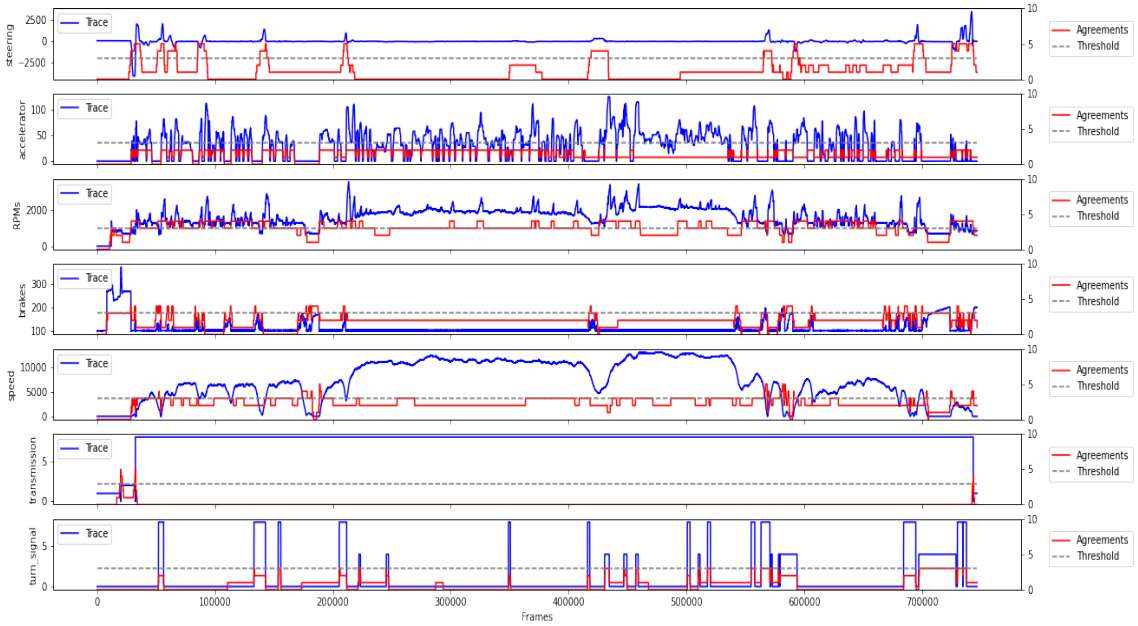


Figure 17: File 02 with Combined Annotation Truth.

The above figure shows where the individual trace behavior has the most agreement. Across every file, the highest numbers are typically around the discrete jumps in the transmission and turn signal traces. The lowest number of agreements occur in the analog channels with the most activity (e.g. accelerator and RPMs). The rest of the files and agreements are shown in Appendix D.

Comparing the individual participants' original annotations to this combined truth demonstrates an expected improvement in the metrics (since the combined traces are constructed from the original annotations). The "wisdom of the crowds" F1-score, Cohen's kappa, annotation error, and Hausdorff distance are shown in Table 14, Table 15, and with

average values improving to 0.4963 (F1-score), 0.4313 (Cohen's kappa), and 50 (annotation error).

Table 14: Combined Annotation Truth, F1-Score.

F1		File Number										
		02	03	04	05	06	09	10	11	12	13	
Analyst Number	01	0.2083	0.2321	0.6342	0.3957	0.5152						0.3971
	02		0.2931	0.6725	0.7211	0.5386	0.6421					0.5735
	03			0.4811	0.8975	0.7712	0.8898	0.8836				0.7846
	04				0.9396	0.8720	0.8716	0.9670	0.5915			0.8483
	05					0.2251	0.2562	0.4263	0.2610	0.8052		0.3948
	06						0.2214	0.1905	0.1539	0.7173	0.1738	0.2914
	07	0.6150						0.1154	0.2506	0.4384	0.2788	0.3396
	08	0.5831	0.4359						0.1915	0.4635	0.4001	0.4148
	09	0.6462	0.3334	0.2282						0.5512	0.7480	0.5014
	10	0.6602	0.5059	0.3582	0.0935						0.4717	0.4179
Mean		0.5426	0.3601	0.4748	0.6095	0.5844	0.5762	0.5166	0.2897	0.5951	0.4145	0.4963
StdDev		0.1892	0.1102	0.1864	0.3593	0.2518	0.3234	0.3915	0.1743	0.1604	0.2185	0.1863

Table 15: Combined Annotation Truth, kappa.

Kappa		File Number										
		02	03	04	05	06	09	10	11	12	13	
Analyst Number	01	0.1568	0.2095	0.6124	0.3710	0.5042						0.3708
	02		0.2566	0.6473	0.7077	0.5253	0.6301					0.5534
	03			0.4545	0.8934	0.7661	0.8865	0.8723				0.7746
	04				0.9370	0.8684	0.8677	0.9642	0.5732			0.8421
	05					0.1887	0.2157	0.3516	0.2175	0.7523		0.3452
	06						0.1791	0.0906	0.1005	0.6431	0.0540	0.2135
	07	0.4915						0.0087	0.2067	0.2967	0.1776	0.2362
	08	0.4275	0.3164						0.1401	0.2919	0.2899	0.2932
	09	0.4954	0.1840	0.1229						0.3033	0.7268	0.3665
	10	0.5106	0.4056	0.2738	0.0192						0.3796	0.3178
Mean		0.4164	0.2744	0.4222	0.5857	0.5705	0.5558	0.4575	0.2476	0.4575	0.3256	0.4313
StdDev		0.1486	0.0890	0.2233	0.3873	0.2642	0.3427	0.4405	0.1883	0.2227	0.2554	0.2197

Table 16: Combined Annotation Truth, Annotation Error.

Error		File Number										
		02	03	04	05	06	09	10	11	12	13	
Analyst Number	01	52	15	0	0	0						13
	02		15	0	0	0	0					3
	03			0	0	0	0	0				0
	04				0	0	0	0	2			0
	05					1	1	1	1	15		4
	06						0	0	2	16	51	14
	07	15						27	29	32	0	21
	08	21	27						20	12	23	21
	09	44	8	4						15	51	24
	10	35	6	8	18						26	19
Mean		33	14	2	4	0	0	6	11	18	30	12
StdDev		15.4	8.2	3.6	8.0	0.4	0.4	12.0	12.9	8.0	21.5	9.3

5.1.2.5 Baseline Annotation Discussion

The expert annotation study collects the baseline data to compare against the autonomous event detections. Several findings are presented below with some exploratory analysis about the root causes.

1. **Low Overall Inter-Coder Agreement Overall-Potentially Contributing Analysis Errors (RQ 1)**

The first observation is that the inter-coder agreement metrics are quite low, well below the 80% suggested by Miles and Huberman [58]. The low agreement is also shown in the number of annotation errors. The technical, subjective nature of the domain led to an expectation of less agreement, however baselines of about 0.21 F1-score and 0.13 (little to none) kappa agreement were lower than anticipated. This discrepancy may be

due to differing opinions on what constitutes an event, different annotation philosophies, ambiguous directions, or unfamiliarity with the proxy dataset provided. Though the true root cause may not be provable, further analysis was performed to better understand this poor performance.

Using File 01 as an example, the F1-score was calculated for each trace individually, shown in Table 17.

The first point to note is the presence of not-a-number (nan) representations from divide by zero errors. Either from analysis decisions or process errors, this means that an participant did not place any event markers in that trace (further supported by the annotation time series shown in Appendix C - Analyst 09, File 01). This is exceptionally present in the brakes, accelerator, and RPMs traces–the most active analog parameters. Though not as prevalent, other likely errors included missing a "close" marker causing all following markers to switch the binary marker incorrectly (e.g. Appendix C - Analyst 07, File 10). On the other hand, there is much better agreement in the other channels–e.g. steering, transmission, and turn signal–with some participant pairings reaching close to perfect agreement. Assuming at least some of the missing annotations are erroneous, possibly due to a failure to correctly save a file, the actual overall agreement may be higher than the coarse metrics strictly shown in the expert annotation study.

For this research there was no attempt to determine participant annotation intent, i.e. error or intentional lack of a marking. Reengaging with participants to fix annotation files, creating a more robust sample from the participant population, choosing a method to remove problem files, or creating more explicit instructions are outside of the limits of the study, but are recommended for future studies of this type.

Table 17: File 01 Trace-Wise F1-Score.

F1	Truth Labels										
Steering	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	1.0000	0.7980	0.5881	0.7117	0.8783	0.6346	0.6977	0.1765	0.0000	0.7926	0.5864
2	0.7980	1.0000	0.6346	0.7865	0.8172	0.7174	0.7749	0.2044	0.0000	0.9150	0.6276
3	0.5881	0.6346	1.0000	0.6283	0.6190	0.5602	0.5103	0.1266	0.0000	0.6845	0.4835
4	0.7117	0.7865	0.6283	1.0000	0.7366	0.6871	0.7383	0.0983	0.0000	0.8475	0.5816
5	0.8783	0.8172	0.6190	0.7366	1.0000	0.6735	0.7409	0.2256	0.0000	0.8136	0.6116
6	0.6346	0.7174	0.5602	0.6871	0.6735	1.0000	0.6139	0.1788	0.0000	0.7404	0.5340
7	0.6977	0.7749	0.5103	0.7383	0.7409	0.6139	1.0000	0.1595	0.0000	0.7971	0.5592
8	0.1765	0.2044	0.1266	0.0983	0.2256	0.1788	0.1595	1.0000	0.0000	0.1821	0.1502
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	nan	0.0000	0.0000
10	0.7926	0.9150	0.6845	0.8475	0.8136	0.7404	0.7971	0.1821	0.0000	1.0000	0.6414
Mean	0.5864	0.6276	0.4835	0.5816	0.6116	0.5340	0.5592	0.1502	0.0000	0.6414	0.4775

Accelerator	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	0.0000
2	0.0000	1.0000	0.1510	0.0000	0.0000	0.0000	0.0000	0.1250	0.0000	0.2188	0.0550
3	0.0000	0.1510	1.0000	0.0000	0.0000	0.0000	0.0000	0.0203	0.0000	0.0271	0.0220
4	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	0.0000
5	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	0.0000
6	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	0.0000
7	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	0.0000
8	0.0000	0.1250	0.0203	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.6385	0.0871
9	nan	0.0000	0.0000	nan	nan	nan	nan	0.0000	nan	0.0000	######
10	0.0000	0.2188	0.0271	0.0000	0.0000	0.0000	0.0000	0.6385	0.0000	1.0000	0.0983
Mean	0.0000	0.0550	0.0220	0.0000	0.0000	0.0000	0.0000	0.0871	0.0000	0.0983	0.0262

Brakes	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
2	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
3	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
4	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
5	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
6	nan	nan	nan	nan	nan	nan	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.3387	0.1113	0.3558	0.0895
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3387	1.0000	0.3344	0.9380	0.1790
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1113	0.3344	1.0000	0.3458	0.0879
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3558	0.9380	0.3458	1.0000	0.1822
Mean	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0895	0.1790	0.0879	0.1822	0.0539

RPMs	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	nan	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	nan	0.0000	0.0000
2	0.0000	1.0000	0.5889	0.0000	0.0000	0.2191	0.0014	0.0013	0.0000	0.0188	0.0922
3	0.0000	0.5889	1.0000	0.0000	0.0000	0.0801	0.0009	0.0013	0.0000	0.0082	0.0755
4	nan	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	nan	0.0000	0.0000
5	nan	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	nan	0.0000	0.0000
6	0.0000	0.2191	0.0801	0.0000	0.0000	1.0000	0.0249	0.0016	0.0000	0.1207	0.0496
7	0.0000	0.0014	0.0009	0.0000	0.0000	0.0249	1.0000	0.8036	0.0000	0.3236	0.1283
8	0.0000	0.0013	0.0013	0.0000	0.0000	0.0016	0.8036	1.0000	0.0000	0.1038	0.1013
9	nan	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	nan	0.0000	0.0000
10	0.0000	0.0188	0.0082	0.0000	0.0000	0.1207	0.3236	0.1038	0.0000	1.0000	0.0639
Mean	-0.1111	0.0922	0.0755	0.0000	0.0000	0.0496	0.1283	0.1013	-0.1111	0.0639	0.0288

Speed	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	1.0000	0.0905	0.0987	0.0844	0.0000	0.4833	0.3028	0.3666	0.1888	0.2643	0.2088
2	0.0905	1.0000	0.9872	0.9183	0.0000	0.5584	0.1946	0.0039	0.4248	0.0231	0.3556
3	0.0987	0.9872	1.0000	0.9245	0.0000	0.5630	0.2101	0.0047	0.4295	0.0249	0.3603
4	0.0844	0.9183	0.9245	1.0000	0.0000	0.5533	0.2064	0.0130	0.4243	0.0206	0.3494
5	0.0000	0.0000	0.0000	0.0000	nan	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.4833	0.5584	0.5630	0.5533	0.0000	1.0000	0.1624	0.3003	0.3934	0.2287	0.3603
7	0.3028	0.1946	0.2101	0.2064	0.0000	0.1624	1.0000	0.0666	0.0782	0.0591	0.1422
8	0.3666	0.0039	0.0047	0.0130	0.0000	0.3003	0.0666	1.0000	0.3532	0.7598	0.2076
9	0.1888	0.4248	0.4295	0.4243	0.0000	0.3934	0.0782	0.3532	1.0000	0.2929	0.2872
10	0.2643	0.0231	0.0249	0.0206	0.0000	0.2287	0.0591	0.7598	0.2929	1.0000	0.1859
Mean	0.2088	0.3556	0.3603	0.3494	0.0000	0.3603	0.1422	0.2076	0.2872	0.1859	0.2457

Transmission	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	1.0000	0.0952	0.1401	0.1062	0.0688	0.0023	0.2168	0.0938	0.0683	0.3703	0.1291
2	0.0952	1.0000	0.0204	0.0128	0.0005	0.0001	0.2818	0.0294	0.0102	0.0415	0.0547
3	0.1401	0.0204	1.0000	0.9382	0.7419	0.0435	0.1062	0.0120	0.7416	0.3905	0.3483
4	0.1062	0.0128	0.9382	1.0000	0.7661	0.0402	0.0612	0.0135	0.7499	0.2554	0.3271
5	0.0688	0.0005	0.7419	0.7661	1.0000	0.0395	0.0281	0.0081	0.9715	0.2473	0.3191
6	0.0023	0.0001	0.0435	0.0402	0.0395	1.0000	0.0017	0.0003	0.0427	0.0160	0.0207
7	0.2168	0.2818	0.1062	0.0612	0.0281	0.0017	1.0000	0.0397	0.0683	0.2290	0.1148
8	0.0938	0.0294	0.0120	0.0135	0.0081	0.0003	0.0397	1.0000	0.0081	0.0236	0.0254
9	0.0683	0.0102	0.7416	0.7499	0.9715	0.0427	0.0683	0.0081	1.0000	0.3197	0.3311
10	0.3703	0.0415	0.3905	0.2554	0.2473	0.0160	0.2290	0.0236	0.3197	1.0000	0.2104
Mean	0.1291	0.0547	0.3483	0.3271	0.0000	0.0207	0.1148	0.0254	0.3311	0.2104	0.1561

Turn Signal	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	1.0000	0.0000	0.1076	0.1313	0.1323	0.0000	0.0000	0.1150	0.0183	0.1306	0.0706
2	0.0000	nan	0.0000	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	######
3	0.1076	0.0000	1.0000	0.8878	0.9138	0.0000	0.0000	0.9120	0.1418	0.9133	0.4307
4	0.1313	0.0000	0.8878	1.0000	0.9575	0.0000	0.0000	0.9529	0.1474	0.9569	0.4482
5	0.1323	0.0000	0.9138	0.9575	1.0000	0.0000	0.0000	0.9899	0.1446	0.9962	0.4594
6	0.0000	nan	0.0000	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	0.0000
7	0.0000	nan	0.0000	0.0000	0.0000	nan	nan	0.0000	0.0000	0.0000	0.0000
8	0.1150	0.0000	0.9120	0.9529	0.9899	0.0000	0.0000	1.0000	0.1426	0.9904	0.4559
9	0.0183	0.0000	0.1418	0.1474	0.1446	0.0000	0.0000	0.1426	1.0000	0.1445	0.0821
10	0.1306	0.0000	0.9133	0.9569	0.9962	0.0000	0.0000	0.9904	0.1445	1.0000	0.4591
Mean	0.0706	0.0000	0.4307	0.4482	0.0000	0.0000	0.0000	0.4559	0.0821	0.4591	0.1947

2. Highest Annotation Errors are From the Least Experienced Analysts (RQ 1)

The second note shows that the two analysts with the least experience, Analyst 07 and Analyst 08 based on pre-survey responses, had an order-of-magnitude more event markers (though interestingly no correlated increase in annotation time). These participants had low performance on the other metrics, but were similar as some other participants, likely due to the high segmentation of having many events.

In the case of these participants, the extra annotations are likely associated with any increase or decrease in the analog parameters and less associated with the stated target detections–i.e. turns, lane changes, deceleration to stop, acceleration from stop, and engine turn-on. The over-annotation may also be associated with less knowledge about expectations and norms for the domain that are shared by the others.

Though these lower performers affect the overall baseline metrics, they are important to be included to capture how new participants may perform with the autonomous system. Lack of domain knowledge or event extraction skills may lead to over- or under-confidence in the autonomous detections. Similarly, performance against these benchmarks may lead to more focused development for trainers.

3. No Other Apparent Correlation Between Pre-Survey and Agreement Performance (RQ 1)

Beyond the novice participant relationships discussed above, there does not seem to be a strong connection between pre-survey responses and the inter-coder agreement metrics. Experience (Question 1.1), confidence in one's skills (Question 1.2), confidence in

machine-learning (Question 1.5), and perceived difficulty of the automated task (Question 2.1) are all poor indicators of inter-coder agreement and annotation time. Though most coders are internally consistent with their time and agreement performance, some vary widely between files. For example, Analyst 06's performance against the combined expert annotations has an F1-score which varies between 0.1539 and 0.7173. These points further demonstrate a volatility among coders which may not be a stable foundation to compare the performance of the autonomous method.

4. **Hausdorff Distance Was Not A Useful Metric in the Presence of Analysis Errors (RQ 1)**

The same analysis errors that led to incorrect event indicator trace creation and poor inter-coder agreement, especially missed "close" markers, likely led to the excessively long distances. With average times between 1 and 5 minutes, these metrics would not be useful for analysis or to compare model performance.

5.2 Parameter Sweep

Concurrently accomplished with the Part 1 case study, the researcher performed a coarse search of hyperparameters to select a model architecture for use in the final automation used in Part 2. This enumerated all options presented in Section 4.3 using the Analyst 01 set (File 01 through File 06) for training data. The researcher self-annotated File 07 to use as a baseline target detection for the model outputs. Training time and AUC were used as evaluation metrics which are enumerated in Table 18 and Table 19.

Table 18: Parameter Sweep Area-Under-the-Curve.

		Autoencoder						Transformer													
		CNN			RNN-LSTM			T2V 32 DFF 1024				T2V 16 DFF 512				T2V 64 DFF 1024				T2V 64 DFF 512			
Window Shape		32x16x32	64x32x64	64x32x16x32x6	32x16x32	64x32x64	64x32x16x32x6	Attention 1	Attention 2	Attention 3	Attention 4	Attention 1	Attention 2	Attention 3	Attention 4	Attention 1	Attention 2	Attention 3	Attention 4	Attention 1	Attention 2	Attention 3	Attention 4
64x8	Trace	0.853	0.7577	0.6773	0.8613	0.8319	0.8388	0.6501				0.699				0.5877				0.7232			
	System	0.6392	0.5885	0.6154	0.6176	0.5865	0.5877	0.6527	0.7064	0.6251	0.648	0.6469	0.6609	0.6705	0.6644	0.6512	0.6753	0.6023	0.6474	0.651	0.6496	0.5659	0.6031
64x16	Trace	**0.8984**	0.8424	0.6721	0.8566	0.8435	0.835	0.825				0.8346				0.6803				0.8152			
	System	0.7352	0.6888	0.7458	0.7165	0.6456	0.6731	0.7491	0.7123	0.7188	0.7098	0.7465	0.7146	0.6772	0.681	0.7495	0.7307	0.6974	0.6138	0.7521	0.7065	0.651	0.6095
64x32	Trace	0.807	0.7898	0.8103	0.8385	0.8381	0.7957	0.7718				0.8206				0.7854				0.764			
	System	0.7791	0.7784	0.7931	0.7952	0.8055	0.7012	0.7788	0.7825	0.743	0.7526	0.7813	0.7487	0.7441	0.7954	0.7804	0.7616	0.7556	0.6625	0.7813	0.7991	0.7108	0.7704
64x64	Trace	0.7656	0.7661	0.7704	0.7708	0.7467	0.8026	0.7542				0.7442				0.7508				0.7755			
	System	0.7785	0.8164	0.7894	0.7683	0.7675	0.7688	0.7384	0.7903	0.7418	0.7724	0.7385	0.7274	0.6989	0.7631	0.7376	0.7372	0.7279	0.7379	0.7382	0.7598	0.7001	0.7369
128x16	Trace	0.7724	0.7485	0.7499	0.8532	0.8599	0.8556	0.7967				0.7174				0.7941				0.6286			
	System	0.7264	0.6748	0.703	0.758	0.6569	0.7165	0.7497	0.7131	0.6579	0.7314	0.7449	0.7024	0.7065	0.6825	0.7478	0.7952	0.7038	0.7338	0.7437	0.6569	0.6941	0.7066
128x32	Trace	0.8036	0.819	0.7575	0.811	0.8202	0.8418	0.7423				0.8264				0.8445				0.8266			
	System	0.8082	0.8	0.7761	0.8109	0.8045	0.8317	0.7959	0.7127	0.7306	0.7944	0.7962	0.7564	0.7538	0.7403	0.7971	0.7237	0.7422	0.7471	0.7987	0.7052	0.771	0.7825
128x64	Trace	0.796	0.7472	0.7475	0.7959	0.7933	0.7177	0.7412				0.8249				0.7169				0.8244			
	System	0.8024	0.7863	0.7726	0.7596	0.7738	0.8056	0.7433	0.789	0.6972	0.7468	0.7426	0.7297	0.739	0.7596	0.7433	0.7262	0.7189	0.7539	0.7438	0.7694	0.6691	0.7415
128x128	Trace	0.7002	0.6845	0.7152	0.7326	0.7098	0.7387	0.7415				0.7415				0.7224				0.6921			
	System	0.7715	0.7318	0.7213	0.7681	0.7737	0.735	0.6435	0.6487	0.6883	0.7148	0.6508	0.7022	0.6093	0.7218	0.6428	0.7661	0.6779	0.6758	0.6431	0.6919	0.6696	0.6583
256x32	Trace	0.6021	0.558	0.6012	0.5958	0.5955	0.5605	0.5329				0.5336				0.5687				0.6102			
	System	0.6192	0.612	0.5911	0.5922	0.6033	N/A	0.5613	0.5829	0.574	0.589	0.5615	0.5722	0.5583	0.5846	0.5649	0.5594	0.5411	0.5575	0.5645	0.5547	0.5606	0.5226
256x64	Trace	0.4905	0.4899	0.5531	0.5808	0.5716	0.5731	0.58				0.5703				0.5386				0.5937			
	System	0.5899	0.6191	0.5827	0.5831	0.5602	0.5185	0.5051	0.5176	0.4827	0.4356	0.5002	0.5217	0.5052	0.497	0.5033	0.5106	0.4779	0.4696	0.5098	0.491	0.494	0.4849
256x128	Trace	0.4651	0.4739	0.5505	0.4996	0.5177	0.4937	0.5245				0.4367				0.5212				0.516	0.516	0.516	0.516
	System	0.5326	0.5743	0.5683	0.4605	0.6214	N/A	0.3895	0.4835	0.3836	0.3596	0.393	0.3716	0.4257	0.3583	0.3909	0.4841	0.3815	0.3322	0.3874	0.3481	0.4632	0.389
256x256	Trace	0.5038	0.4965	0.5002	0.533	0.5197	0.5352	0.4849				0.5041				0.4553				0.5385			
	System	0.5306	0.5187	0.5586	0.5889	0.4806	0.5149	0.4849	0.4048	0.3718	0.3985	0.415	0.4162	0.369	0.3724	0.3843	0.5235	0.3351	0.3616	0.4331	0.4329	0.3167	0.343

Table 19: Model Training Time (minutes).

	Autoencoder						Transformer			
	CNN			RNN-LSTM			T2V 32	T2V 16	T2V 64	T2V 64
Window Shape	32x16x32	64x32x64	64x32x16x32x64	32x16x32	64x32x64	64x32x16x32x64	DFF 1024	DFF 512	DFF 1024	DFF 512
64x8	5	6	7	16	18	26	23	17	30	25
64x16	2	3	3	8	9	13	12	8	15	12
64x32	1	2	2	4	5	7	6	5	8	6
64x64	1	1	1	2	2	3	3	2	4	3
128x16	3	4	5	14	15	24	20	13	26	21
128x32	2	2	2	7	8	12	10	7	13	11
128x64	1	1	1	4	4	6	5	3	7	5
128x128	0	1	1	2	2	3	3	2	3	3
256x32	3	3	4	14	15	23	20	13	27	22
256x64	1	2	2	7	8	12	12	7	21	10
256x128	1	1	2	4	6	9	5	3	7	5
256x256	0	0	1	2	2	3	3	2	3	3

For most of the tested models, the AUC varied between 0.74 and 0.85 until the window shape reached 256 where the model performance dropped precipitously to 0.40 to 0.58.

A few more key observations are noted below.

1. Window Size and Stride Generally Has Little Effect on Performance

Several input data shapes were tested by varying the window size and stride when applied to the raw time series data. For both the 64 and 128 sample window length, there was little variation in the AUC at the different levels of overlap. However, there was a significant decrease in performance when tested at 256 samples. This is likely due to the models' ability to accurately reconstruct the data when the windows are shorter—at least for the size of this dataset. With more volume of representative data, the models may be able to model the input more accurately and increase performance for larger windows.

2. System-Wide Detections Underperform Trace-Wise

One of the methods used to determine events was to compare aggregated expert annotations, flattened versions of the trace-wise labels, against the events detected by latent space distances. In almost every case, the models using the individual traces yielded a higher AUC than the system-wide metric. This may be due to the more direct relationship between the original input data and the trace-wise reconstruction compared to the abstraction of the system-wide metrics. Similarly, the latent space distances may not be as representative of the underlying data.

3. Shallower Autoencoder Architectures Perform Better

The third observation from Table 18 shows simpler autoencoders, fewer hidden layers and fewer nodes, generate a better AUC for both CNNs and RNNs. This is could be caused by the small dataset. With larger, more complicated AEs, the models are better able to memorize the input, overfitting the data, creating less reconstruction error and latent space distance between samples at inference. This will reduce the generalizability of the models when applied to new data.

5.2.1 Model Selection and Rationale

A convolutional architecture with the 64x16 window size and 32x16x32 shape was selected. In this particular use case, the AUC of 0.8984 was the highest among tested models. Additionally, a training time of only 2 minutes over 100 epochs will be sufficiently timely to present to participants in Part 2. Figure 18 shows the selected model architecture and Figure 19 shows the model performance.

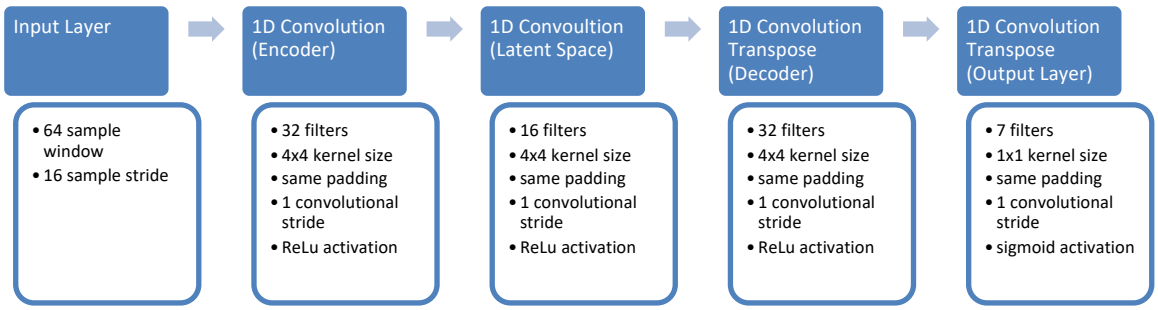


Figure 18: Convolutional Autoencoder Architecture

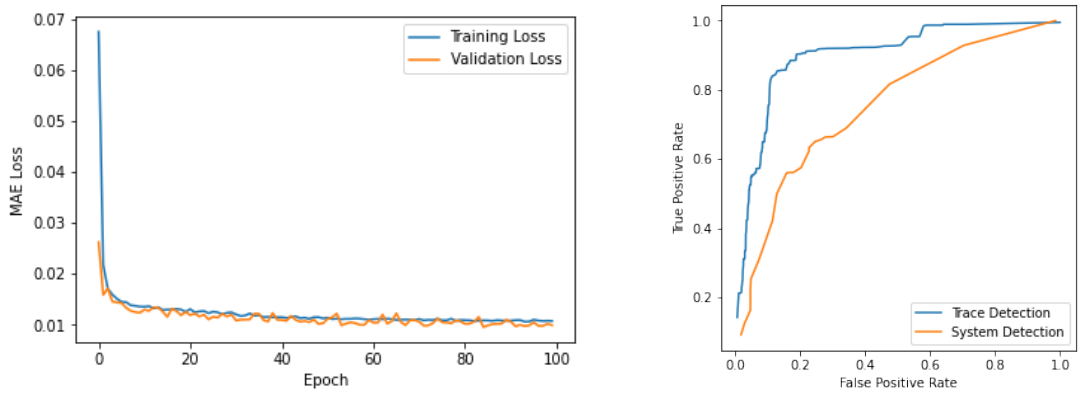


Figure 19: Training Error (left) and ROC (right) for Selected CNN Autoencoder.

After the model was selected, mean-based and histogram-based static thresholding methods were tested on the reconstruction error and first-differenced traces. Using the researcher generated File 07 as the test file again, the researcher performed another coarse search testing the mean-based method from 0 to 3 standard deviations from the mean and the histogram-based method at 10, 100, and 1000 bins. The detections were observed when the researcher tested with individually selected thresholds for each trace. The performance was qualitatively estimated based on the researcher's domain expertise and visual inspection.

The histogram binning method with 10 bins was chosen on the first-differenced error trace. Target error bins were selected for each signal and are summarized in Table 20.

Table 20: Histogram Bins for Reconstruction Error Thresholds.

	Bin Number
Steering	1
Accelerator	4
Brakes	2
RPMs	2
Speed	1
Transmission	2
Turn Signal	3

Lastly, multiple blur values were tested on this system and a value of 32 was chosen based on visual inspection of the output ROC curves.

This method for optimization search was not particularly rigorous for this case study. Ideally, a combined expert annotations from the expert annotators would have been used for this testing. However, difficulties arose with data transfer to allow participants to operate in their natural settings. This necessitated an initial model search to be performed concurrently and offline from the rest of the case study. It was determined that facilitating participant ease and tool familiarity was more important for the research goals, e.g. testing delivered utility and analytic acceleration, than providing a truly optimized model architecture.

As a final summary, the selected model is a CNN autoencoder that uses trace-wise, first-differenced reconstruction error as its metric. A histogram with 10 bins is calculated using the training data and a bin selection array shown in Table 20. If the error metric exceeds the value of the bin, it is considered an event.

5.3 Part 2 Results

As shown in the pre-survey discussion, human-in-the-loop correction of autonomous outputs was considered an important step for survey participants, especially when trying to establish the overall reliability of the system before implementation. However, the poor inter-coder agreement results in the expert annotation study indicate that this is a difficult task.

Part 2 was designed to answer all three research questions associated with the hypothesis. This was done by giving case study participants the opportunity to run the autonomous event detection method themselves and then capture their perceptions of the interaction. The post-detection corrections study gives participants the opportunity to grade and critique the autonomous detections. Finally, post-survey records participant thoughts about the program in its current form. Generally, the domain experts think the autonomous method has promise, however too many events were detected and they spent more time removing them than the time spent to initially label the data in Part 1.

Analyst 08 was unable to participate in Part 2. Analyst 10 did not correct File 12.

5.3.1 Automation Assisted Annotation Results

The third step in this case study was the automation assisted portion. This evaluates the autonomous event detection against the combined expert annotation traces created in pre-survey and participant-corrected versions of the model outputs. This is done using the same inter-coder agreement metrics used for the expert annotations study. For each participant, this step trains the CNN autoencoder using the data files that a participant annotated in Part 1 (but does not use those annotations because this is an unsupervised

method). This creates a uniquely trained CNN autoencoder for each participant's EDAS model. Then, the remainder of the files from a participant's case study data plan is passed through the autoencoder and first-differenced threshold to generate event detections. Participants are asked to correct these detection files based on their domain expertise and record their time required to finish the task. Part 2 uses File 07 as its baseline for comparing inter-coder agreement for the correction files.

Generally, the Part 2 evaluation metrics were similar to those in Part 1.

5.3.1.1 File 07 - All Analysts Inter-Coder Correction Agreement

Similar to File 01 in Part 1, a baseline for the post-detection, participant corrections were established using File 07. Since each participant in Part 2 trains and test the model with different data, resulting in ten unique models, the output event detections are not a direct comparison. Each participant begins from a different starting point events file. However, the resultant corrections should be similar if the coders agreed on the events.

Table 21 show the F1-score and Cohen's kappa for the post-detection corrections study.

Table 21: Inter-coder Correction Agreement for Part 2, F1-score.

F1		Truth Labels										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	1.0000	0.2802	0.5361	0.4643	0.4912	0.1228	0.4551	N/A	0.1368	0.1504	0.2930
	02	0.2802	1.0000	0.3477	0.2221	0.2338	0.1058	0.1936	N/A	0.1373	0.1030	0.1804
	03	0.5361	0.3477	1.0000	0.3006	0.5083	0.0842	0.2944	N/A	0.0873	0.0769	0.2484
	04	0.4643	0.2221	0.3006	1.0000	0.2757	0.1025	0.6815	N/A	0.2364	0.2722	0.2839
	05	0.4912	0.2338	0.5083	0.2757	1.0000	0.0749	0.2751	N/A	0.1623	0.1256	0.2385
	06	0.1228	0.1058	0.0842	0.1025	0.0749	1.0000	0.1268	N/A	0.0845	0.6131	0.1461
	07	0.4551	0.1936	0.2944	0.6815	0.2751	0.1268	1.0000	N/A	0.1758	0.2745	0.2752
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	0.1368	0.1373	0.0873	0.2364	0.1623	0.0845	0.1758	N/A	1.0000	0.4158	0.1596
	10	0.1504	0.1030	0.0769	0.2722	0.1256	0.6131	0.2745	N/A	0.4158	1.0000	0.2257
Mean		0.2930	0.1804	0.2484	0.2839	0.2385	0.1461	0.2752	N/A	0.1596	0.2257	0.2279

Table 22: Inter-coder Correction Agreement for Part 2, kappa.

Kappa		Truth Labels										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	1.0000	0.2422	0.5200	0.4251	0.4675	0.0411	0.4165	N/A	0.0567	0.0658	0.2483
	02	0.2422	1.0000	0.3249	0.1644	0.1977	0.0211	0.1359	N/A	0.0559	0.0121	0.1282
	03	0.5200	0.3249	1.0000	0.2706	0.4925	0.0393	0.2648	N/A	0.0427	0.0301	0.2205
	04	0.4251	0.1644	0.2706	1.0000	0.2303	-0.0679	0.6448	N/A	0.0925	0.1128	0.2081
	05	0.4675	0.1977	0.4925	0.2303	1.0000	0.0040	0.2311	N/A	0.0984	0.0548	0.1974
	06	0.0411	0.0211	0.0393	-0.0679	0.0040	1.0000	-0.0248	N/A	-0.2986	0.3785	0.0103
	07	0.4165	0.1359	0.2648	0.6448	0.2311	-0.0248	1.0000	N/A	0.0338	0.1312	0.2037
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	0.0567	0.0559	0.0427	0.0925	0.0984	-0.2986	0.0338	N/A	1.0000	0.0701	0.0168
	10	0.0658	0.0121	0.0301	0.1128	0.0548	0.3785	0.1312	N/A	0.0701	1.0000	0.0950
Mean		0.2483	0.1282	0.2205	0.2081	0.1974	0.0103	0.2037	N/A	0.0168	0.0950	0.1476

In the above Tables, the inter-coder agreements are slightly higher when using the automation than without (Part 1 annotations). The F1-score varies between 0.0749 and 0.6815 (mean 0.2279) with kappa values between -0.2986 and 0.6448 (mean 0.1476). These metrics are still low showing little agreement between participants.

The second set of comparative metrics are the annotation errors, shown in Table 23, and the Hausdorff distances, in Table 24.

Table 23: Inter-coder Correction Agreement for Part 2, Annotation Error.

Error		Truth Labels										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels	01	0	3	1	1	26	3	3	N/A	15	14	7
	02	3	0	2	4	29	0	6	N/A	18	17	9
	03	1	2	0	2	27	2	4	N/A	16	15	8
	04	1	4	2	0	25	4	2	N/A	14	13	7
	05	26	29	27	25	0	29	23	N/A	11	12	20
	06	3	0	2	4	29	0	6	N/A	18	17	9
	07	3	6	4	2	23	6	0	N/A	12	11	7
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	15	18	16	14	11	18	12	N/A	0	1	12
	10	14	17	15	13	12	17	11	N/A	1	0	11
Mean		7	9	8	7	20	9	7	N/A	12	11	10

The annotation errors were better than those in Part 1 showing a lower difference between the number of events found by different participants. These values varied between 0 and 29 errors (mean 10). Hausdorff distances also performed better. After removing a few outliers at 497,936 samples, the metric varied between 16 and 228,400 (0.0351 seconds to 9 minutes) with means between 2,818 and 32,343 samples (6.184 seconds to 1.183 minutes). The distances and time are much narrower in dynamic range but will still only be useful in offline applications (in the CAN analysis domain).

Table 24: Hausdorff Distance for Part 2 Correction, File 07.

Hausdorff	Truth Labels										
Steering	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	2612	2038	3162	2410	3146	3206	2144	3212	2620	2134	2668
2	4206	2710	4040	3816	5247	6117	3538	4632	4102	3249	4166
3	2504	3666	2568	3038	3070	3556	2500	3160	3012	2564	2964
4	2147	3413	2230	2118	2262	3727	2246	2803	2783	2310	2604
5	1888	192	1456	2096	2144	1984	2144	1872	1712	1764	1725
6	2667	3005	3216	2507	3200	2518	3033	3323	2731	2771	2897
7	2436	1263	2500	2096	2468	2727	2377	1872	2500	1436	2168
8	0	0	0	0	0	0	0	0	0	0	0
9	50795	1152	1104	50779	50859	86608	49995	1872	1E+05	464	44152
10	3424	3800	3488	3264	3639	4047	3607	4080	3184	2359	3489
Mean	7268	2124	2376	7212	7604	11449	7158	2683	17053	1905	7426
Accelerator	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	2432	2288	2048	608	2064	640	317	0	1616	2064	1408
2	2432	5613	5453	2400	2064	1568	2144	0	1056	13961	3669
3	2080	240	0	2400	1104	1568	2144	0	0	1744	1128
4	2432	2288	2048	1826	2064	502	1746	0	1360	4829	1910
5	1712	1040	1056	1374	1104	1305	1200	0	1024	1734	1155
6	2432	2288	2048	2400	2064	1568	2144	0	1056	2064	1806
7	2432	2288	2048	1104	2064	1072	450	0	1616	2064	1514
8	0	0	0	0	0	0	0	0	0	0	0
9	1E+05	1E+05	1E+05	7481	1E+05	528	16077	0	1E+05	1E+05	63783
10	1E+05	1E+05	1E+05	5305	1E+05	1750	1E+05	0	1E+05	1E+05	72266
Mean	21920	22035	21901	2490	21578	1050	13075	0	21420	23171	16515
Brakes	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	722	1712	976	608	626	2016	1952	1008	1232	976	1183
2	1328	1712	1744	1408	1280	2208	2976	1872	1232	1296	1706
3	1328	1712	1744	1408	1280	2208	2976	1872	1232	1296	1706
4	1328	1712	1152	943	1280	2208	2976	1872	1232	1264	1597
5	1072	1712	1744	1088	0	2016	1952	1024	1232	1088	1293
6	2E+05	2E+05	2E+05	5E+05	2E+05	2E+05	1E+05	2E+05	2E+05	2E+05	2E+05
7	848	848	976	512	864	832	2138	896	880	880	967
8	0	0	0	0	0	0	0	0	0	0	0
9	1280	672	1744	224	192	1968	12483	1024	0	1104	2069
10	81568	82094	82062	22903	82142	55392	38336	32759	82046	82046	64135

Mean	Truth Labels										
	24301	34147	32054	52082	25675	23698	17219	21153	33841	26917	32343
Test Labels 1	0	1504	176	1280	1056	768	1104	736	1088	112	782
2	1264	2112	1152	2128	2064	1808	1920	1952	1728	1184	1731
3	1216	2112	1152	2128	2064	1808	1920	1952	1728	112	1619
4	2163	1504	2195	1235	1315	1395	1443	1427	1427	451	1456
5	848	2239	880	952	757	757	575	112	952	112	818
6	1264	2112	1152	2128	2064	1808	1920	1952	1728	1184	1731
7	1264	1296	440	1280	1088	667	619	635	1136	208	863
8	0	0	0	0	0	0	0	0	0	0	0
9	1120	13296	1024	1104	2180	1024	10472	224	0	1184	3163
10	15328	13389	15456	6992	14480	14624	12032	15824	7552	16282	13196
Mean	2447	3956	2363	1923	2707	2466	3201	2481	1734	2083	2818
Speed	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	10300	5426	10620	10140	6484	10140	16396	5620	10140	10140	9541
2	22817	5186	22785	23233	2464	3024	23249	10490	9776	23233	14626
3	3216	4425	4752	4176	2464	3024	3184	3760	4864	4832	3870
4	106768	1E+05	93872	1E+05	1E+05	93408	97088	1E+05	93408	92384	1E+05
5	2832	2272	2672	2640	1849	2800	2016	3760	3120	2672	2663
6	9840	5714	9920	9632	5522	4738	9936	10512	9776	9920	8551
7	91031	88359	88791	98055	98727	88343	88871	10432	88343	87303	82826
8	0	0	0	0	0	0	0	0	0	0	0
9	3216	1984	18608	22352	2464	22352	16736	11312	165151	23040	28722
10	106768	1E+05	93872	1E+05	1E+05	93408	97088	1E+05	93408	99264	1E+05
Mean	35679	32021	34589	37647	35252	32124	35456	26558	47799	35279	39156

Transmission	Truth Labels										
	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	0	32	1104	1424	1104	16	32	16	1040	477
2	600	600	584	600	1024	600	601	600	601	600	641
3	32	32	0	32	1424	32	1120	16	1120	1040	485
4	0	0	32	0	1024	0	16	32	16	64	118
5	144	144	176	320	1024	320	304	176	304	320	323
6	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05
7	16	16	32	16	1024	16	0	48	0	64	123
8	0	0	0	0	0	0	0	0	0	0	0
9	16	16	32	16	1024	16	0	48	0	64	123
10	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05	5E+05
Mean	99668	99668	99679	99796	1E+05	99796	99793	99682	99793	99906	1E+05
Turn Signal	1	2	3	4	5	6	7	8	9	10	
Test Labels 1	0	0	0	0	0	64	0	0	1280	0	134
2	1088	1088	1088	1088	1088	1088	1088	1088	1280	1088	1107
3	0	0	0	0	0	64	0	0	1280	0	134
4	0	0	0	0	0	64	0	0	1280	0	134
5	0	0	0	0	0	64	0	0	1280	0	134
6	1088	1088	1088	1088	1088	1088	1088	1088	1280	1088	1107
7	0	0	0	0	0	64	0	0	1280	0	134
8	0	0	0	0	0	0	0	0	0	0	0
9	94480	94480	94480	94480	94480	95456	94480	94480	1E+05	94480	96596
10	5680	5680	5680	5680	5680	6338	5680	5680	6338	5680	5812
Mean	10234	10234	10234	10234	10234	10429	10234	10234	12996	10234	11699

5.3.1.2 Inter-Coder Correction Agreement Across Other Files

After the expert collections study, all study participants have either annotated or corrected every file in the case study data plan (except File 08 which was excluded). Table 25 summarizes the average values.

Table 25: Summary of Part 2 Inter-coder Correction Agreement.

	Baseline	File Number									
	07	02	03	04	05	06	09	10	11	12	13	Mean
F1-Score	0.2279	0.2244	0.2053	0.2460	0.1639	0.1958	0.2260	0.1486	0.2731	0.2923	0.3091	0.2284
kappa	0.1476	0.0587	0.0544	0.1566	0.0493	0.0484	0.1514	-0.0039	0.1045	0.1356	0.2438	0.0999
Error	10	28	29	11	42	36	18	25	21	55	86	35

The metrics depicted above show similar bounds and means near those of the Part 1 annotations and the Part 2 baseline in File 07. Again, these values show little agreement between participants.

5.3.1.3 Correction Time Analysis

Research Question 3 is that the automation will accelerate event extraction. The time required to both run the event detection interface and correct the model output was collected then compared against the initial time taken to annotate each file. This is summarized in Table 26 (File 07 is compared against File 01 for the baselines).

Table 26: Correction Times for Part 2.

		File Number													
File Time (mins)		01	02	03	04	05	06	07	08	09	10	11	12	13	
		26.45	27.38	20.25	3.77	20.48	18.45	19.77	27.55	14.88	14.57	12.83	22.33	22.95	
Participant Number	01		-	-	-	-	-	1.265		2.016	1.030	1.169	1.343	0.654	1.246
	02		1.424	-	-	-	-	0.658		-	0.961	1.169	1.567	1.264	1.174
	03		1.096	1.235	-	-	-	2.124		-	-	1.714	2.194	1.307	1.612
	04		1.023	0.889	3.979	-		1.113		-	-	-	1.567	0.654	1.537
	05		1.278	0.741	2.653	0.732	-	0.809		-	-	-	-	0.697	1.152
	06		0.950	0.593	2.387	0.732	1.030	0.809		-	-	-	-	-	1.083
	07		-	1.383	4.509	1.270	1.247	2.934		0.874	-	-	-	-	2.036
	08		-	-	N/A	N/A	N/A	N/A		N/A	N/A	-	-	-	N/A
	09		-	-	-	0.977	0.813	1.416		1.344	1.030	1.169	-	-	1.125
	10		-	-	-	-	2.547	2.276		6.452	2.265	1.403	N/A	-	2.989
Mean		N/A	1.154	0.968	3.382	0.928	1.409	1.489	N/A	2.671	1.321	1.325	1.668	0.915	1.550
StdDev		N/A	0.194	0.332	1.024	0.255	0.779	0.784	N/A	2.563	0.630	0.240	0.366	0.339	1.584
% Increase		N/A	0.477	-0.084	0.045	0.218	1.407	0.626	N/A	3.321	0.504	0.164	1.780	-0.139	0.466

The time required to correct a file, on average, was between 0.915 and 3.382 minutes per minute of data. For the most part, it took longer to correct the model outputs than simply to annotate the original file, however File 03 (the short file) and File 13 saw minor improvements in speed. Overall, these percent increases ranged from -0.139 to 3.321 (mean 0.466).

Based on the case study data plan, the participants likely started with File 07 then the next file numerically (wrapping around to File 02 after File 13). Similar to the expert annotation study, there does not appear to be any time savings acceleration while correcting subsequent files.

5.3.1.4 Autonomous Detections Versus Participant Corrections

By comparing the system outputs with participant corrections, objective metrics are calculated about how much the participant agrees or disagrees with the autonomous detections. Generally, participants agreed with events found in the discrete channels but

disagreed with many of the analog channel annotations. This is further supported qualitatively by post-survey responses (discussed below).

First using File 07, Table 27, Table 28, and Table 29 shows the inter-coder agreements for the detections and corrections treating the humans as the truth providers.

Table 27: Corrections versus Detections, F1-score.

F1		Truth Labels (EDAS)										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels (Participant)	01	0.6600	0.4247	0.5768	0.6419	0.4846	0.5666	0.5076	0.4826	0.5474	0.5556	0.5448
	02	0.1276	0.0828	0.1447	0.2106	0.1798	0.2368	0.0898	0.1495	0.1920	0.1091	0.1523
	03	0.4619	0.4803	0.5239	0.5272	0.5482	0.3970	0.3626	0.5290	0.3868	0.4654	0.4682
	04	0.2832	0.2041	0.3102	0.3863	0.2405	0.3596	0.2888	0.2641	0.3400	0.3049	0.2982
	05	0.4767	0.4797	0.4782	0.5112	0.5944	0.4051	0.4195	0.4896	0.4107	0.4640	0.4729
	06	0.0558	0.0340	0.0559	0.0688	0.0537	0.0948	0.0759	0.0573	0.0731	0.0490	0.0618
	07	0.3003	0.2153	0.3044	0.3358	0.2312	0.3655	0.4158	0.2655	0.3501	0.3072	0.3091
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	0.1061	0.0809	0.1118	0.1216	0.0991	0.1493	0.1171	0.0858	0.1686	0.1026	0.1143
	10	0.0828	0.0568	0.0840	0.1000	0.0621	0.1137	0.1040	0.0634	0.1105	0.0905	0.0868
Mean		0.2838	0.2287	0.2878	0.3226	0.2771	0.2987	0.2646	0.2652	0.2866	0.2720	0.2787

Table 28: Corrections versus Detections, kappa.

Kappa		Truth Labels (EDAS)										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels (Participant)	01	0.6466	0.4080	0.5600	0.6262	0.4676	0.5450	0.4852	0.4659	0.5255	0.5382	0.5268
	02	0.0928	0.0561	0.1105	0.1758	0.1525	0.1984	0.0480	0.1219	0.1526	0.0740	0.1183
	03	0.4464	0.4686	0.5103	0.5126	0.5369	0.3767	0.3424	0.5174	0.3666	0.4502	0.4528
	04	0.2469	0.1766	0.2752	0.3510	0.2098	0.3157	0.2457	0.2352	0.2967	0.2702	0.2623
	05	0.4577	0.4655	0.4593	0.4917	0.5820	0.3785	0.3956	0.4743	0.3851	0.4447	0.4534
	06	-0.0001	-0.0030	-0.0004	0.0045	0.0103	0.0173	0.0078	0.0156	-0.0022	-0.0065	0.0043
	07	0.2658	0.1887	0.2700	0.2986	0.2006	0.3234	0.3814	0.2372	0.3088	0.2735	0.2748
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	0.0532	0.0457	0.0589	0.0611	0.0578	0.0767	0.0522	0.0454	0.1012	0.0504	0.0603
	10	0.0260	0.0196	0.0270	0.0347	0.0176	0.0330	0.0343	0.0205	0.0338	0.0351	0.0282
Mean		0.2484	0.2029	0.2523	0.2840	0.2483	0.2516	0.2214	0.2370	0.2409	0.2366	0.2424

Table 29: Corrections versus Detections, Annotation Error.

Error		Truth Labels (EDAS)										
File 07		01	02	03	04	05	06	07	08	09	10	
Test Labels (Participant)	01	6	35	20	26	33	65	35	20	56	23	32
	02	69	58	67	55	40	84	94	53	75	74	67
	03	32	25	30	32	21	87	59	22	72	41	42
	04	19	28	17	17	26	74	42	17	59	28	33
	05	72	67	80	74	59	123	99	72	108	79	83
	06	61	50	59	57	44	94	88	39	85	68	65
	07	44	71	42	40	57	43	25	62	44	47	48
	08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	09	83	108	83	81	98	46	76	103	47	82	81
	10	50	77	66	64	73	89	59	72	82	63	70
Mean		48	58	52	50	50	78	64	51	70	56	58

The first table shows the humans actually agree more with the EDAS detections, on average, than they do with each other. F1-scores vary between 0.0568 and 0.6600 (mean 0.2787) with kappa scores from -0.0065 to 0.6466 (mean 0.2424). Lastly, there were between 6 and 123 annotation errors (mean 58).

Table 30 summarizes the mean values across the other files and further illustrates similar performance.

Table 30: Summary of Other Correction-Detection Metrics.

	File Number										
	02	03	04	05	06	09	10	11	12	13	Mean
F1	0.1588	0.1448	0.4304	0.2078	0.1640	0.2210	0.1708	0.1531	0.4337	0.3066	0.2391
kappa	0.1137	0.1038	0.3500	0.1616	0.1104	0.1904	0.1150	0.1043	0.2608	0.2716	0.1782
Error	77	57	25	100	76	45	60	48	290	73	85

In the post-survey, the participants discussed the need for significant corrections to model outputs. This is supported by the low agreement scores between what was originally flagged as events and how the participants believed the events should be represented.

5.3.1.5 Participant Corrections Versus Combined Expert Annotations

Next, a comparison is made between the corrections and the combined expert annotations data created in the expert annotations study. This shows how the participants perform against a notional, more accurate "wisdom of the crowds" events trace.

Table 31 and

Table 32 show the F1-score and Cohen's kappa for the corrections and combined expert annotation events.

Table 31: Corrections versus Combined Annotations, F1-score.

F1		File Number - Combined Truth										
		02	03	04	05	06	09	10	11	12	13	
Participant Number	01						0.3764	0.1687	0.0682	0.2699	0.3619	0.2490
	02	0.2592						0.4257	0.1675	0.3804	0.2064	0.2878
	03	0.1486	0.1060						0.5562	0.0866	0.4996	0.2794
	04	0.2387	0.4054	0.3812						0.2614	0.5785	0.3730
	05	0.5306	0.1118	0.4485	0.2553						0.3143	0.3321
	06	0.0972	0.1318	0.2040	0.1759	0.1514						0.1521
	07		0.3108	0.1629	0.1159	0.0850	0.3419					0.2033
	08			N/A	N/A	N/A	N/A	N/A				N/A
	09				0.0792	0.0745	0.1515	0.1065	0.1315			0.1086
	10					0.0560	0.1127	0.2245	0.0786	N/A		0.1180
Mean		0.2549	0.2132	0.2992	0.1566	0.0917	0.2456	0.2314	0.2004	0.2496	0.3921	0.2335

Table 32: Corrections versus Combined Annotations, kappa.

Kappa		File Number - Combined Truth										
		02	03	04	05	06	09	10	11	12	13	
Participant Number	01						0.3505	0.0804	0.0131	-0.0506	0.2511	0.1289
	02	-0.0483						0.3778	0.1156	0.1977	0.1346	0.1555
	03	0.0755	-0.0344						0.5396	-0.2037	0.4636	0.1681
	04	0.0282	0.3460	0.3256						0.0315	0.5386	0.2540
	05	0.3710	-0.0782	0.4018	0.2232						0.2536	0.2343
	06	-0.1188	-0.0060	0.1008	0.1096	0.1111						0.0393
	07		0.2094	0.0624	0.0521	0.0460	0.3126					0.1365
	08			N/A	N/A	N/A	N/A	N/A				N/A
	09				0.0079	0.0290	0.1041	-0.0338	0.0768			0.0368
	10					0.0068	0.0586	0.1002	0.0167	N/A		0.0456
Mean		0.0615	0.0874	0.2227	0.0982	0.0482	0.2065	0.1312	0.1524	-0.0063	0.3283	0.1330

When compared against the combined traces, the corrections perform on par with the inter-coder agreement metrics for the initial annotation task. The F1-score varies between 0.0560 and 0.5785 (mean 0.2335) and the kappa ranges from -0.2037 to 0.5396 (mean 0.1330). This shows internal consistency between the manual tasks in Part 1 and Part 2. However, these agreements are still quite low and well below a level considered reliable.

The annotation errors in Table 33 also show how the performance metrics are similar to the initial annotation task performed in expert annotations study. These values range from 0 to 52 (mean 18) and are concentrated in the analog parameters. Most of these errors occur in File 02 and File 13. Excluding those poor performers, the errors instead vary between 1 and 34.

Table 33: Corrections versus Combined Annotations, Annotation Error.

Error		File Number - Combined Truth										
		02	03	04	05	06	09	10	11	12	13	
Participant Number	01						3	13	0	3	46	13
	02	51						0	1	16	51	24
	03	49	11						1	3	45	22
	04	45	11	7						15	44	24
	05	26	1	12	26						15	16
	06	52	15	0	0	0						13
	07		9	20	34	30	22					23
	08			N/A	N/A	N/A	N/A	N/A				N/A
	09				13	21	20	11	23			18
	10					21	19	12	19	N/A		18
Mean		45	9	10	18	18	16	9	9	9	40	18

5.3.1.6 Autonomous Detections Versus Combined Expert Annotations

The final comparison represents the performance of this unsupervised system without human interaction. The combined expert annotations is a measure of how a collective group could have annotated certain files. Treating them as unknown target detections, and comparing it against the autonomous detections, shows how this method

could perform while deployed in an actual analytic setting. The F1-score, Cohen's kappa, and annotation errors are shown in Table 34, Table 35, and Table 36.

Table 34: Detections versus Combined Annotations, F1-score.

F1		File Number										
		02	03	04	05	06	09	10	11	12	13	
Training Set	01						0.2711	0.1575	0.2049	0.2297	0.2266	0.2180
	02	0.0767						0.1381	0.2013	0.2721	0.1274	0.1631
	03	0.1063	0.1718						0.2432	0.2765	0.2548	0.2105
	04	0.1173	0.2104	0.3679						0.2453	0.2666	0.2415
	05	0.0879	0.0891	0.3230	0.1911						0.1275	0.1637
	06	0.1750	0.1853	0.3739	0.2165	0.3348						0.2571
	07		0.1873	0.2299	0.1612	0.1797	0.1984					0.1913
	08			0.3907	0.2484	0.2637	0.3890	0.1632				0.2910
	09				0.2085	0.1870	0.2537	0.1611	0.2142			0.2049
	10					0.1597	0.2417	0.1421	0.1868	0.1438		0.1748
Mean		0.1126	0.1688	0.3371	0.2051	0.2250	0.2708	0.1524	0.2101	0.2335	0.2006	0.2116

Table 35: Detections versus Combined Annotations, kappa.

Kappa		File Number - Combined Truth										
		02	03	04	05	06	09	10	11	12	13	
Training Set	01						0.2479	0.1158	0.1762	-0.0186	0.1848	0.1412
	02	0.0418						0.1037	0.1813	0.0666	0.0937	0.0974
	03	0.0582	0.1326						0.2186	0.0304	0.2121	0.1304
	04	0.0636	0.1652	0.3055						-0.0094	0.2180	0.1486
	05	0.0501	0.0600	0.2695	0.1630						0.0887	0.1263
	06	0.1026	0.1500	0.3111	0.1845	0.3144						0.2125
	07		0.1488	0.1611	0.1254	0.1549	0.1731					0.1527
	08			0.3386	0.2272	0.2466	0.3743	0.1278				0.2629
	09				0.1728	0.1585	0.2260	0.0973	0.1828			0.1675
	10					0.1323	0.2168	0.0839	0.1584	-0.0384		0.1106
Mean		0.0633	0.1313	0.2772	0.1746	0.2013	0.2476	0.1057	0.1835	0.0061	0.1595	0.1550

In the above tables, the F1-score varies between 0.0767 and 0.3907 (mean 0.2116) and kappa between -0.0384 and 0.3743 (mean 0.1550). Though the max performance for F1-score and kappa are lower than the annotations, the dynamic ranges are much narrower and the overall means are near the original annotations (0.2116 and 0.1550, respectively).

These numbers are also very close to the correction performance against the combined expert annotations, discussed above. This shows a greater stability across the different files. However, the original annotations included the likely analysis errors which contributed to lower scores.

Table 36: Detections versus Combined Annotations, Annotation Error

Error		File Number - Combined Truth										
		02	03	04	05	06	09	10	11	12	13	
Training Set	01						3	5	2	28	45	17
	02	35						7	3	59	40	29
	03	48	11						0	4	44	21
	04	46	10	8						16	44	25
	05	41	8	6	14						39	22
	06	44	12	11	9	5						16
	07		12	6	6	1	2					5
	08			3	3	2	1	3				2
	09				3	3	4	6	1			3
	10					3	1	1	1	8		3
Mean		43	11	7	7	3	2	4	1	23	42	14

The annotation errors range from 0 to 59 (mean 14). Again, these errors occur in File 02 and File 13 but File 12 was included as well. Without them, the errors instead vary between 1 and 14. Similar to the other annotation error discussions, the qualitative analysis of an acceptable number of errors would be strictly domain and application dependent and is out-of-scope for this thesis. It should be noted that previous comparisons included a human for correction purposes, so a truly autonomous system would likely need to be held to higher standards than one working alongside an participant.

5.3.1.7 Post-Detection Corrections Study Discussion

The post-detection corrections study collected quantitative measures to empirically show the performance and utility of an unsupervised anomaly detection method. With the low agreement metrics across every comparative test, there is obviously room for improvement. Below are a few additional observations about the post-detection corrections study.

1. **Changing Threshold Levels Can Dramatically Change Performance (RQ 1)**

The final note to mention is that the thresholds established during the parameter sweep search have a significant impact on the event detection performance, especially for parameters with significant variation and activity. Figure 20 shows an example signal and the associated reconstruction error and first-difference trace. The seemingly stochastic nature of the metric traces shows that a static threshold can have difficulty accounting for certain behaviors. Small changes to the selected histogram bins lowered or raised the thresholds directly leading to fewer or more events, respectively. Since these levels were set by the researcher, this method is not completely autonomous. Further investigation into better bin selection, more sophisticated thresholding methods (e.g. dynamic thresholding), or simply leaving the bin selection to the user may each increase performance.

2. **Detections Perform Similarly to Single-Participant Corrections (RQ 1)**

The second observation is that the humans and system performed similarly when compared against the combined expert annotations traces established by the other

participants in Part 1. The participants performed slightly better with an average of 0.2335 F1-score over the deep learning method's 0.2116. There were participants who performed much better than the computer, reaching up to 0.5785, but also some who underperformed down to 0.0560. These highs and lows, like the initial annotations, generally track with experience, though some typically high performers could still have a much lower value on a single file. Conversely, the autonomous method has a narrower range with fewer outliers from the mean. The detections also had fewer annotation errors in general.

Though the humans did not agree with the computer across many event markers, as evidenced by the Section 5.3.1.4 discussion, they did not perform well against the same benchmark either (on average). Participants may have been steered by the poor initial markers or ambiguous directions (post-survey responses), but their lack of agreement between each other in Sections 5.3.1.1-2 and the expert annotations study suggest it is difficult to get the domain experts to agree.

3. Certain Files Appear More Difficult, Show Poor Detections and Annotations (RQ 2)

As noted above, File 02, File 12, and File 13 have more annotation errors when compared against the expert truth. Looking at the model-derived detections, Figure 20, the analog parameters are exceptionally over-marked.

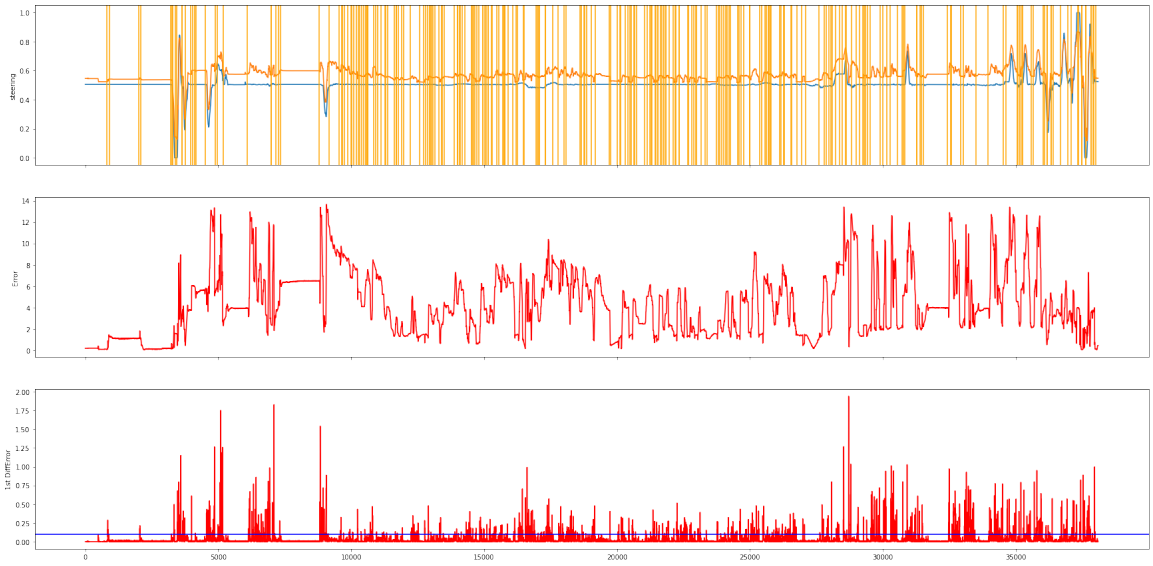


Figure 20: Steering Data, Error, and First-Difference Traces (with Threshold)

There are no obvious differences in the data, however they all share File 06 and File 09 as training data. Looking at the signal traces for these two files (Appendix D), there is a small difference in brake activity during engine turn-on than is typically observed in the data (caused by a second driver). This could have impacted the autoencoder outputs in early timestamps, however later samples should not be affected because CNN inputs are independent. Regardless, it seems the model overfits the training set which causes greater reconstruction errors in problem files. Training on more data, both in volume and more diverse activity representation, may help normalize the performance across all the test files.

4. Empirical Analysis Shows No Time Savings When Including Participant Correction (RQ 3)

Saving participant time was a key goal for this research project–which was not met. Due to complications with the natural setting, models had to be trained during the post-detection corrections study. For each participant, the CNN autoencoder was trained at

about 2-3s per epoch (3.3 to 5.0 minutes) and the rest of the program took around 5 minutes to run leading to a total initial investment of 10 minutes–a task typically performed offline from analysis. It was the correction times that took the longest.

Some increase in time for Part 2 was expected. Participants are accomplishing two tasks: validation and correction. However, an average 30% increase is significant. As noted before (and later during post-survey analysis), participants disagreed with many of the automated event detections, especially in the analog parameters like the accelerator, RPMs, and brake signals. This led to many corrections or deleting all markers and starting over. Additionally, there was a loss of trust in the system across all traces due to the poor performance in others.

Regardless of the reasons for the slower pace, it is apparent that this method does not deliver time savings when an participant needs to make such significant corrections.

5.3.2 Post-Survey Results

The post-survey in post-survey was the final interaction with the domain experts. This survey instrument was used to query and record their interactions and perceptions of the event detection method as is–simplicity of the method, the reliability of the detections, the applicability to their analytic domain–along with the factors which increased and decreased their trust. Table 37 shows the survey results.

Table 37: Post-Survey Results.

		Participant Number									
	Question	01	02	03	04	05	06	07	08	09	10
1.1	How would you characterize the ease of the event detection method?	3	4	4	3	3	2	3	N/A	4	1
1.2	Characterize your approximate time and effort savings factoring in both program run-time and output validation.	1	4	1	3	3	1	2	N/A	1	2
1.3	How did your thinking about automated event extraction change after running this method?	SHORT ANSWER									
2.1	How confident are you in your validation assessment of the autonomously extracted events?	2	3	4	3	4	1	2	N/A	2	1
2.2	How useful do you believe this autonomous event extraction method could be for your application domain?	2	4	4	2	3	1	3	N/A	1	5
2.3	How would you characterize the trustworthiness of the event outputs for this CAN domain application? (i.e. Would you allow this program to present you with candidate events without re-validation?)	1	3	1	1	2	1	1	N/A	1	1
2.4	What factors increased your trust? (Short Answer	SHORT ANSWER									
2.5	What factors decreased your trust?	SHORT ANSWER									

5.3.2.1 Likert-like Scale Questions

Like the pre-survey, Likert-like questions were used for participants to respond on a specified scale. A few observations are noted below.

1. **Event Detections Were Not Trustworthy (RQ 2)**

Similar to the point above, the large number of corrections probably led participants to mark the method as ‘not trustworthy’ or ‘minimally trustworthy’ (Question 2.3). The analysts were also ‘not confident’ or ‘minimally confident’ with their correction assessments of the autonomously detected events. This could be due to the large number of corrections which reduced their trust and confidence in their own methods since they were at such odds with the computer process. The lack of confidence and perceived trustworthiness of EDAS directly addresses research question 2, in the negative.

2. **The System Did Not Have Perceived Savings (RQ 3)**

Research question 3 addresses whether EDAS saves compute, time, or burden from the manual task. Supported by empirical data presented above, participants did not perceive any time savings (Question 1.2). Coupled with their answers to Question 1.1 and Question 2.3, there were no apparent savings in effort either. This likely came from perceived need for significant corrections to detections output by the autonomous system (discussed further in the short answer section).

3. **Experts Believe the Method Could Still Be Useful in their Domains**

Despite the perceived issues with the system, most participants believe the method could have at least a 'minimally useful' application to their domain with some participants marking higher usefulness.

5.3.2.2 Short Answer Questions

The post-survey short answers for Question 1.3, Question 2.4, and Question 2.5 reinforced the general method perceptions shown in the Likert-like scale questions. Namely, there was not a time savings and the autonomous system was not helpful. These questions also asked about what increased or decreased the participant trust in the model outputs, possibly leading to subsequent research or participant education. Additional observations on the short answer questions are discussed below.

1. **Too Many Activities Were Flagged as Detections, Some Appeared Random (RQ 2)**

The primary factor which changed participants' thoughts about the method (Question 1.3) and lowered trust (Question 2.5) were that too many events were detected. Most participants noted that discrete parameters, i.e. turn signal and transmission, were flagged correctly, however the performance was not good for continuous parameters "place events in a way that appeared almost random". Another participantstated, "I am uncertain if the distribution of event markers would have been noticeably different… if event markers were assigned by mapping a random number generator against the timestamps". For some participants, they stated it was easier to delete all detections and annotate it all over.

2. **Instructions Were Not Sufficiently Clear to be Confident with Results (RQ 2)**

The final point noted by survey participants was that the instructions for event annotations and corrections were vague. The domain experts would have preferred more explicit directions about what constituted an event. One participant noted, "I struggled through marking up the files initially as I could think of 3-20 different ways to construct events." and "... I had difficulty telling if the script was attempting to use my understanding of an event…" Likely a function of the highly analytic demographic, this may have led to frustrations which impacted confidence in both the system outputs and human corrections. However, this is the target demographic and accommodating these participants is necessary

for meaningful research. Clearer, more explicit instructions should be considered in future research.

5.3.2.3 Post-Survey Discussion

Most participants' expectation of some usefulness from this method, despite its stated issues, further reinforces the need for automated event detection in these domains. The dramatic over-detection for continuous parameters, and the resulting lack of trust by participants, represents a significant challenge for this method (in its current form). However, additional comments about unclear, "frustratingly vague" instructions and seemingly random detections may be possible to overcome with education—though it is unknown if this conjecture is true or not.

First of all, the instructions about what exactly constituted an event were intentionally ambiguous–the participants were only instructed what types of events to look for. From the researcher's target domain knowledge, different participants have strong and differing opinions about what traces to include and how to mark an event. This was intended to be captured in the initial annotations and participants corrections, based on their current domain knowledge. Some better inter-coder agreement may have been by explicitly directing annotation guidelines, but that would have impacted the participants in their natural setting, and possibly added unintended bias to the study. The system did require pairs of event markers which meant, according to one participant, "...I was not able to mark events and/or regions in the fashion I would normally." This was an unfortunate artifact of the method which further led to frustration. Better definition for how to attribute and/or

read autonomous events will be necessary for real domain application, but that is beyond the scope of this thesis.

This system clearly struggles with the highly-active, seemingly random activity in some of the continuous traces. Frustratingly, these are also less indicative of a particular activity class than the discrete parameters. The accelerator position, for example, is active in every target event (except braking) and the operator usage is constantly changing. This causes many detections. These are not random as noted by the participants, but reacting to an activity that an participant would typically not attribute to an event.

Similarly, participants mentioned, “Often there were event markers in the steering wheel position when there was absolutely no change whatsoever in the value of that channel. It seemed that these events may have been associated with activity in the other channels…” These types of events, and many of the other seemingly erroneous detections, are likely contextual events caused by a lack of change in a channel. For example, an increase or decrease in RPMs is typically associated with the start and end of a turn. During engine turn-on, there is a quick spike in RPMs without associated steering activity, which creates reconstruction error in the steering channel, and is flagged as an abnormal event. Like the conclusions of pre-survey, education about how deep neural nets actually find events may increase trust in this method.

5.4 Final Discussion

The purpose of this research was to deliver time savings and trusted autonomous event detections to domain experts then record their perceptions before and after interaction

with this new method. Ultimately, this autonomous event detection method, executed in this particular domain and natural analytic setting, were unsuccessful in reaching these research goals. Though the participant domain experts did not express any perceived savings, and empirical evidence supported their assertions of poor event detection performance, there were several key findings which will inform subsequent research. These are discussed below.

1. Participants Did Not Agree on How to Annotate Events (RQ 1)

The first key finding was that different participants strongly disagreed with each other on their definitions of events. They selected different representative signals, bounding around perceived activity (Hausdorff distance), and varied greatly on the total number of events (annotation error). This occurred with novice and senior experience, confident and not confident participants, and in both Part 1 and Part 2 extraction tasks. Though this may have stemmed from misunderstandings associated with the study directions, having such a high-level of disagreement makes it difficult to establish an objective benchmark to compare the accuracy, reliability, and stability of different event detection methods.

However, there was generally better agreement among Participants 01-05. These participants are from the same office, and all have the same domain expertise and CPS analysis targets as the researcher. A similar mental model and/or common analytic experiences/training may have contributed to similar annotations and corrections.

Regardless, the strong overall disagreement calls into question the validity of the entire experiment and all resulting conclusions reached about from this research should be

considered suspect. It also represents the long-term difficulty of satisfying diverse participants' needs when trying to accelerate their analysis processes.

2. This Application Does Well with Discrete Parameters, Poorly with Analog (RQ 1)

Noted by both case study participants and the quantitative research, the CNN autoencoder and histogram-based static threshold did well with the discrete parameters but struggled with analog signals. Unfortunately, events in discrete channels are easier for both humans and simpler detection methods to identify. As one participant noted, "...these events are not difficult to extract automatically at all…"

One potential problem with this autonomous detection method is the static thresholding technique. For simplicity and to simulate a human-out-of-the-loop system, the same thresholding method was applied over every file with identical histogram bin selections. For discrete parameters, the jump in value from one state to another created large, short duration reconstruction errors and were easily discovered by the static method. High variability and activity of the analog signals created a more stochastic behavior in the original trace, reconstruction error, and first-difference. This led to seemingly random and excessive periods that exceeded the chosen thresholds–at least as perceived by the participants. More sophisticated thresholding methods may improve performance, however analog parameters may already be less indicative of target behaviors. For example, distinguishing the difference between a left and right turn is less concerned with the RPMs, speed, accelerator, and brakes. The turn signal (and steering) are more indicative for that type of activity. In the original model and thresholding selection, the analog parameters

already have higher histogram bins. Assigning even higher thresholds to flag an event, and focusing more on discrete parameters, may also improve performance.

3. Participants Have Difficulty with Contextual Anomalies (RQ 1)

During the post-survey discussion, a comment from a domain expert about event detections with no discernible activity is likely associated with contextual anomalies/events. The absence of activity being anomalous is more likely to be missed by an participant during normal analysis and is, therefore, less likely to be accounted for in legacy methods. It is common for AI/ML methods to fill a specific, narrow purpose. Looking at autonomous event detection as a technique to help participants, searching for contextual anomalies may be a unique and valuable contribution to the analytic process.

4. More Data Would Likely Improve Performance (RQ 1)

As mentioned above, File 02, File 12, and File 13 have higher annotation errors due to more frequent detections. This may be due to being trained on data from a secondary driver, File 06 and File 09. The overall dataset is also small. By collecting and training on more data, the models may be more generalizable from incorporating more examples of driving activities.

To create cross-validation metrics, this method also does not fully utilize autoencoder functionality. It is not uncommon for autoencoders to test on the same data it is trained on because they are intended to reconstruct the input data and the reduction in dimensionality causes the measurable errors/distances for event detection. By training the

CNN autoencoder with all of the data, and testing every file, the system would learn representative activity from every file in the set and possibly perform better during event detection.

5. Detection Method Was Counter Some Participants' Thought Processes (RQ 2, RQ 3)

New methods do not always fit cleanly within existing paradigms. Old methods remain because they work, but they may not evolve to meet the demands of a changing analytic environment. The autonomous event detection method clearly did not fit in the existing analysis domain. It did not increase their inter-coder agreement and it actually increased their annotation time–but this could be due to the tasks being counter to their normal thought processes. One participant said that they do not bound the start and end of most activities (a study directive during annotations). Not accounting for contextual anomalies was another disconnect between the autonomous system and the human process. A general lack of knowledge about AI/ML technologies may have confused participants, so some basic training and education may show immediate increase in trust.

This research was designed to fit the analytic pipeline described in Figure 1 but substituting autonomy of the laborious manual processes. However, a key point was raised when a participant stated, "In creating my own event-marking logic flow, I optimize for eye-friendliness." Similarly, multiple other participants mentioned that there were too many detections to be useful. Optimizing for human-readability does not necessarily optimize for computer-based pattern recognition. Autonomous methods are not applicable for every task, so there may need to be distinct research paths for human-machine teaming

analysis and methods optimized for autonomy (e.g. event detection and follow-on classification).

6. Current Method Insufficient for the Application Domain, Shows Promise (RQ 3)

Though some participants found utility in this autonomous event detection method, "... it is far away from being a stand-alone analysis." In an operational domain, where participant findings drive decision-making, the current system clearly cannot replace legacy methods–but it may be incorporated. It was only meant to be part of the analysis process and shows promise in finding initial candidate events in unfamiliar data or searching for contextual events in complex systems.

5.5 Researcher Performance

Due to process requirements which allowed participants to operate in their natural setting, the researcher had to perform the coarse optimization search and model selection concurrently with the rest of the participant studies. Selecting the model using annotations not provided by participants induced external bias into the subsequent studies. To quantify this bias, the researcher's File 07 annotation file was compared against the EDAS detections (Table 38) and participants' corrections (Table 39).

Table 38: Researcher Annotations versus EDAS Detections, Summary.

	Participant Number (Training Set)										
	01	02	03	04	05	06	07	08	09	10	Mean
F1-Score	0.4296	0.3640	0.4258	0.4720	0.3766	0.3914	0.3892	0.3784	0.4372	0.4133	0.4078
kappa	0.4039	0.3437	0.3999	0.4454	0.3536	0.3556	0.3569	0.3561	0.4053	0.3872	0.3808
Error	26	37	34	40	47	73	51	36	60	25	43

Table 39: Researcher Annotations versus Post-Detection Corrections, Summary.

	Participant Number										
	01	02	03	04	05	06	07	08	09	10	Mean
F1-Score	0.4464	0.1462	0.3370	0.3878	0.3572	0.1041	0.3999	N/A	0.1905	0.1907	0.2844
kappa	0.4120	0.0925	0.3112	0.3301	0.3222	-0.0125	0.3458	N/A	0.0857	0.0748	0.2180
Error	26	59	44	27	76	53	66	N/A	93	48	55

Table 39 shows similar levels average of disagreement with the other participants. Since the EDAS model was selected using Participant 01's training set, it was unsurprising that one of the best inter-coder agreements of the researcher's annotations was with Participant 01.

Table 38 shows that, after the initial model selection, the researcher's annotation file performed better when compared to the EDAS output. On average, the researcher's F1-score had a 278.5% increase over the participants for each of the ten different models trained only on subsets of the data. The metric values were also more stable, only varying between 0.3640-0.4720 for F1-score and 0.3437-0.4454 for Cohen's kappa. This higher level of inter-coder agreement is not surprising since the researcher was familiar with AI/ML applications and the particular behaviors of the EDAS. This further supports the proposition that better education and/or more robust instructions could yield better overall performance. However, these higher metrics are still well below the target 0.80 F1-score with a kappa only considered fair.

Considering the excellent performance using the area-under-the-curve (AUC) metric during the parameter sweep (0.89 AUC), the poor inter-coder agreement performance suggests that the EDAS could have performed better. Perhaps, the thresholds were not well-selected. Additionally, using the entire dataset to train the selected autoencoder better utilizes the offline autoencoder capability and could improve the system performance.

VI. Conclusions and Future Work

Event detection is a difficult problem with a long history of extensive research. In the past, this task has primarily been done with statistical and/or supervised methods, but the size and speed at which data must be analyzed is exponentially increasing. To maintain decision-advantage, we must find novel and powerful ways to draw insight from data–at the speed of relevance. Unsupervised (and/or semi-supervised) methods will likely be a foundational research area to deliver this utility and acceleration.

This research hypothesizes that an unsupervised event detection algorithm consisting of a temporal neural network, error and distance metrics, and a static threshold can deliver demonstrable savings to domain experts in terms of accuracy (F1-score, annotation error, Hausdorff distance) and qualitatively through a case study with domain experts.

To discuss a final conclusion on this research on unsupervised event detection, it is important to reevaluate the stated research questions: 1. delivering autonomous event detections in multivariate time series with unsupervised learning; 2. providing sufficient accuracy and reliability for domain experts to trust the method; and 3. realize savings in time, compute, or analyst burden.

Looking at research question 1, the selected model was successfully able to find some events which matched the human identified events (though with a low F1-score between 0.20 and 0.40). It was especially good at finding changes in the discrete channels and contextual anomalies missed or discounted by the participants. However, both the quantitative analysis and qualitative survey results clearly show that the method did not

meet the other research goals of reliable detections (research question 2) and savings (research question 3).

The CAN domain and the analytic target domain of the participants represent two application domains for this type of research, but they are the ones being tested here. To generalize the CAN analysis presented here to the target domain, it is critical to find representative data to train the algorithms within the participants' natural setting and to perform an extensive optimization search before trust can be built. It will also be important to include additional analytic programs to account for the additional complexities of the target domain. Lastly, good evaluation metrics will be critical. Going forward, it may be more fruitful to first work with the target domain participants to better annotate or define an event. With minimal inter-coder agreement, the humans were not a good, stable baseline to empirically compare against the autonomous system.

This research contributes the event detection automation system (EDAS) with an application to the cyber-physical system (CPS) domain. Additionally, this research ran a case study that recorded and analyzed domain expert participants' perceptions of EDAS as it relates to their actual analysis domain.

Though EDAS was able to autonomously detect events in the CPS time series data, and answered research question 1 affirmatively, EDAS did not provide the perceived accuracy or reliability for participants to develop trust (research question 2) and did not realize any savings from their manual methods (research question 3).

Five areas for Future Work are discussed below.

6.1 Human-Machine Integration Future Work

The first area of Future Work involves additional study of how CPS analysts interact with their analytic domain.

1. Establish Community Agreement for Annotation and Correction Guidelines

CPS event annotations and corrections were shown to have significant variability between domain experts with little agreement. The first Future Work study should attempt to establish community guidelines for annotating CPS time series data. Without an agreed upon method, the analytic community will not be able to improve inter-coder agreement. After having a common baseline to compare machine learning techniques, a metric to test potential annotators before trusting them to deliver intelligence products. The annotations metric can also guide improvements to the autonomous system. Without the annotator metric, inter-coder agreement will remain low. Similarly, trainers could use the metric to improve developmental analysts.

2. Determine If an Autonomous Event Detection Method Is Directly Suited for Human Validation

The second study should focus on whether an autonomous system's interface design that is optimized for computer-based pattern recognition and event detection can also be optimized for the human validation and correction task. As mentioned above, multiple participants felt the way EDAS found and displayed events was counter to their analytic process. The study proposed here would need to test whether the requirements for

event detection tasks can be similar enough to those of the humans' mental model and vision for the human-machine team. This is a key connection to research question 2 and should be studied along with the way these changing requirements affect participant trust.

3. User Experience Study

The last study that should follow the research presented here should focus on the user experience with EDAS or an associated, dedicated user interface. The study would test various levels of user control over the event detection system and ask the research question: In this particular application, would allowing users to have some control over the automation or providing explainability to the event detection marking increase trust? Study variables could include event detection confidence ratings, participant-adjustable thresholds, or reporting inter-coder agreement to the user after a file has been corrected. Lastly, this study should also calculate and record whether the higher-level of user control actually leads to more accurate and reliable event detections or if inter-coder agreement will remain low.

6.2 Machine Learning Future Work

The second set of Future Work suggestions focus on improving the EDAS with additional machine learning research and application.

1. More Complex Networks, Better Optimization Search, More Sophisticated Thresholding

While it is common to state "more data is always better," Nakkiran, et al. [64] argue there are two regions in training deep networks when model complexity is compared against dataset size. When the model is small compared to the sample size, the system is *under-parameterized* and there is a reduction in performance with larger models. However, the authors show there is a critical model complexity where the system can *interpolate* the data and reach near zero training error. After that, test error only decreases as model complexity increases. Therefore, simply testing wider and deeper autoencoders or stacked transformers may increase performance, however this will likely require even more data.

As stated in the parameter sweep, only a small number of hyperparameter settings and architecture shapes were tested. Additionally, only a single file and training set were used for the optimization search. A more comprehensive search of the hyperparameter space may yield better results.

A more sophisticated thresholding method such as the dynamic thresholding method shown in Hundman, et al.'s research [10] may also improve the overall accuracy and reliability of the autonomous method.

2. Ensemble Methods

Next, ensemble methods should be researched. Like the combined expert annotations improved performance on the Part 1 annotations, a "wisdom of the crowds"

approach of training multiple, decoupled models and/or using different thresholding methods could yield better overall results through voting. Ensemble methods already have a presence in legacy machine learning applications [19, pp. 189-212] and in recent literature.

3. Spatio-Temporal Transformers

The attention-based transformer is a new development in sequence-to-sequence modeling, forecasting, and reconstruction. Though typically seen as the cutting-edge research in natural language processing, transformers have also realized performance gains when applied to time series analysis tasks. Spatio-temporal transformers are a research area which may have significant application to the CAN and target domains, where event detection correlation is relevant in both the temporal and cross-channel dimensions. Some examples are found in 3D human motion prediction [40], cross-dimensional geo-tagged measurement imputation [41], and video recognition [42]. Also, transformers may be able to infer longer sequences than the model was trained on without losing significant accuracy [39].

4. Dealing With Multiple Data Rates

Whether it is an internet-of-things application or specifically related to CAN data analysis, it is important to deal with data that has different sampling or transmission rates. For the CAN domain, each data packet is sent based on a set priority and output frequency. This thesis only used a small subset of the total signals on-board the vehicle with a simple

linear interpolation or backfill method to impute data and fill the sparse data matrix. A significant amount of information was discarded to simplify the analysis problem. However, more sophisticated event detection methodologies would need to account for all of this data. Bin´kowski, et al. [65] developed a method using a learnable vector for different measurements from disparate sensors. Other applications exist which use different sized RNN sizes to detect emotion during conversation [32] or CNNs which use skip connections when changing feature map size and depth [19, pp. 466-478], i.e. time series of different lengths.

5. Segmentation Methods

The final research area to explore reframes the event detection problem into a time series segmentation problem. Instead of looking at the timestamps between different activities, these methods would look instead at the characteristics of the periods between the event markers. Like segmenting music into a verse, chorus, or bridge [66] [67] [68], the driving activities would transfer from one state to another. In fact, Elspas, et al. [27] uses a weakly supervised, fully convolutional network to segment driving scenarios. Other methods still include U-Time [30], ClaSP [69], and breakpoint detection [11].

As stated by multiple participants in the post-survey, this method did not meet their expectations in capability or performance. It cannot, and should not, stand on its own in a fully autonomous detection system—"The correct identification of changes in discrete parameters shows that this method has the potential to be useful, but it still needs work."

However, this research was only intended to be a starting point. When Gen. Brown charged the Air Force to "accelerate change," he also realized the importance of "failing forward"—essentially learning from every experience, even unsuccessful projects. This thesis will be a foundation for that additional, future experimentation.

Appendix

Appendix A – HRPP Exempt Determination Form

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (AETC)

HRPP Exempt Determination Form

For AFIT HRPP Use Only	
Protocol Number:	
Protocol Title:	

EDO Determination

Does this submission meet an Exempt Criteria? *Select the appropriate exemption category. Categories are defined in Exemption Request Package and on Page 2 of this form.*

☑ Yes

Which exempt category applies? **32 CFR 219.104 (d) (2)** (i)

Is a limited IRB Review required to determine adequate provisions are in place to protect the privacy of subjects and maintain confidentiality of data? ☐ **Yes** ☑ No

If a limited IRB review is required, IRB Member determined that *either*:

☐ Sufficient measures were taken to protect privacy and confidentiality.

- OR -

☐ Insufficient measures were taken to protect privacy and confidentiality.

☐ No

☐ The human subject research does not meet any exempt criteria. Referred to AFRL IRB Chair for IRB review.

- OR -

☐ The research uses an In Vitro diagnostic device with specimens that are NOT individually identifiable. Referred to AFRL IRB Chair to determine compliance with applicable FDA regulations.

AFIT EDO / IRB Member Submission Analysis

EDO Reviewer Comments

The research seems to be exempt under exempt category 2. Research involves surveys. Any disclosure of the human subjects' responses outside the research would not reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, educational advancement, or reputation, and the researcher is taking steps to safeguard the information.

AFIT EDO Signature

CUNNINGHAM.WILLIAM.A.III.1230804660

Digitally signed by CUNNINGHAM.WILLIAM.A.III.1230804660
Date: 2022.03.21 10:43:27 -04'00'

Click or tap to enter a date.

Exempt Determination Official

Date 21 March 2022

Note: To sign this form electronically, please save it as a PDF and follow these instructions.

20 June 2019 1

Exempt Categories
32 CFR 219.104(d)(1) Exempt Category 1
Research, conducted in established or commonly accepted educational settings that specifically involves normal educational practices that are not likely to adversely impact students' opportunity to learn required educational content or the assessment of educators who provide instruction. This includes most research on regular and special education instructional strategies, and research on the effectiveness of or the comparison among instructional techniques, curricula, or classroom management methods.
32 CFR 219.104(d)(2) Exempt Category 2
Research that only includes interactions involving educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior (including visual or auditory recording) if at least one of the following criteria is met:
(i) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects cannot readily be ascertained, directly or through identifiers linked to the subjects; (ii) Any disclosure of the human subjects' responses outside the research would not reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, educational advancement, or reputation; or (iii) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects can readily be ascertained, directly or through identifiers linked to the subjects, and an IRB conducts a limited IRB review. Complete Section 6.
32 CFR 219.104(d)(3)(i) Exempt Category 3
Research involving benign behavioral interventions in conjunction with the collection of information from an adult subject through verbal or written responses (including data entry) or audiovisual recording if the subject prospectively agrees to the intervention and information collection and at least one of the below criteria are met. Please provide sufficient detail in section 4.1 to ensure the criteria has been met. Please refer to the Investigator Guidance on this topic.
(A) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects cannot readily be ascertained, directly or through identifiers linked to the subjects; (B) Any disclosure of the human subjects' responses outside the research would not reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, educational advancement, or reputation; or (C) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects can readily be ascertained, directly or through identifiers linked to the subjects, and an IRB conducts a limited IRB review. Complete Section 6.
Note: Benign behavioral interventions are brief in duration, harmless, painless, not physically invasive, not likely to have a significant adverse lasting impact on the subjects, and the investigator has no reason to think the subjects will find the interventions offensive or embarrassing. If the research involves deceiving the subjects regarding the nature or purposes of the research, this exemption is not applicable unless the subject authorizes the deception through a prospective agreement to participate in research in circumstances in which the subject is informed that he or she will be unaware of or misled regarding the nature or purposes of the research.

32 CFR 219.104(d)(4) Exempt Category 4
Secondary research for which consent is not required: Secondary research uses of identifiable private information or identifiable biospecimens, if at least one of the following criteria is met:
(i) The identifiable private information or identifiable biospecimens are publicly available; (ii) Information, which may include information about biospecimens, is recorded by the investigator in such a manner that the identity of the human subjects cannot readily be ascertained directly or through identifiers linked to the subjects, the investigator does not contact the subjects, and the investigator will not re-identify subjects; (iii) The research involves only information collection and analysis involving the investigator's use of identifiable health information when that use is regulated under 45 CFR parts 160 and 164, subparts A and E, for the purposes of "health care operations" or "research" as those terms are defined at 45 CFR 164.501 or for "public health activities and purposes" as described under 45 CFR 164.512(b); (HIPAA Regulations)
Note: HIPAA applies and includes either an authorization or waiver of authorization. It does not include bio specimens, only protected health information (PHI).
Note: This does not include primary collection from subjects for the proposed research. It allows both retrospective and prospective secondary use.
32 CFR 219.104(d)(5) Exempt Category 5
Research and demonstration projects that are conducted or supported by a Federal department or agency, or otherwise subject to the approval of department or agency heads (or the approval of the heads of bureaus or other subordinate agencies that have been delegated authority to conduct the research and demonstration projects), and that are designed to study, evaluate, improve, or otherwise examine public benefit or service programs, including procedures for obtaining benefits or services under those programs, possible changes in or alternatives to those programs or procedures, or possible changes in methods or levels of payment for benefits or services under those programs.
Note: These must be posted on a federal website.
32 CFR 219.104(d)(6) Exempt Category 6
Taste and food quality evaluation and consumer acceptance studies, (i) If wholesome foods without additives are consumed or (ii) If a food is consumed that contains a food ingredient at or below the level and for a use found to be safe, or agricultural chemical or environmental contaminant at or below the level found to be safe, by the Food and Drug Administration or approved by the Environmental Protection Agency or the Food Safety and Inspection Service of the U.S. Department of Agriculture.

Benign Behavioral Interventions

Research involving benign behavioral interventions in conjunction with the collection of information from an adult subject through verbal or written responses (including data entry) or audiovisual recording if the subject prospectively agrees to the intervention and information collection and at least one of the following criteria is met:
(A) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects cannot readily be ascertained, directly or through identifiers linked to the subjects;
(B) Any disclosure of the human subjects' responses outside the research would not reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, educational advancement, or reputation; or
(C) The information obtained is recorded by the investigator in such a manner that the identity of the human subjects can readily be ascertained, directly or through identifiers linked to the subjects, and an **IRB conducts a limited IRB review of privacy and confidentiality** to make a determination of exemption. *This determination is based on adequate provisions to protect privacy of subjects and to maintain the confidentiality of the data, based on the July 26, 2017 DHHS (Department of Health and Human Services) and local policies. Data Security Review will likely be required.*

What is a benign behavioral intervention?

- Behavioral interventions must be brief in duration (a few minutes or hours). Although there is no specific amount of time that is defined as brief, OHRP guidance suggests the intervention must be brief in nature, even if subsequent data collection takes longer.
- Interventions may not be harmful, painful or distressing. Risk to subjects is low.
- Interventions must be unlikely to have significant emotional discomfort or adverse lasting impact
- Study content and procedures must not be offensive or embarrassing to subjects
- Medical interventions and procedures are not permissible in this exemption
- Physical (bodily) tasks and physical exercise should not be included in this exempt category.
- Deception can only be used if the subject prospectively agrees to the use of deception. Subjects must be informed prior to initiating the intervention that they will be unaware of, or misled regarding the true nature or purpose of the research. They will also be told whether further information will be provided at the conclusion of the research activities. Researchers should consider de-briefing subjects.
- Research procedures in this exempt category should generally be limited to:
  - communication or interpersonal contact with the subject,
  - the performance of a cognitive, intellectual, educational or behavioral task, or
  - manipulation of the subject's physical, sensory, social, or emotional environment
- Data collection in this exempt category is limited to:
  - verbal (oral) or written responses by the subject
  - data entry by the subject
  - observation of the subject
  - audiovisual recording

This category does not include the introduction or administration of instruments, substances or energy onto or into the body for research data collection. For example: Fitbit, bioharness, eye tracker, EEG, etc…

Some examples benign behavioral interventions:

- Performing cognitive tasks
- Providing educational materials to participants with the intention of changing their behavior
  - (e.g. smoking cessation, eating habits)
- Playing an online game
- Playing economic games
- Being exposed to stimuli such as color, light or sound at safe levels
- Solving puzzles under various noise conditions

Note: If the research involves deceiving the subjects regarding the nature or purposes of the research, this exemption is not applicable unless the subject authorizes the deception through a prospective agreement to participate in research in circumstances in which the subject is informed that they will be unaware of or misled regarding the nature or purposes of the research.

Appendix A.1 – Message to Research Participants

Message:

Part 1: Expert Annotation—----------

Thank you for participating in the user-survey for my thesis. This project attempts to use machine learning methods to accelerate man-in-the-loop processes and autonomously extract 'events' from multi-variate time series data. The automation will present an analyst with candidate events that you will be asked to assess their accuracy and reliability, and if the automation aids your analysis process.

The data you will be analyzing is several collections of controller area network (CAN) data logged from a personally owned and operated vehicle (POV). For ease of analysis, the data has gone through the first pre-analysis format extraction process. The vehicle signals extracted include:

1. Steering column position
2. Accelerator
3. Vehicle speed
4. Engine RPMs
5. Brakes
6. Transmission Position (i.e. Park, Reverse, Neutral, Drive, 1, 2)
7. Turn Signal

Part 1)

You will first be asked to complete a pre-survey to characterize your experience with analyzing time series data and perceptions of the event extraction problem and prior automations.

You will then perform event extraction on six (6) data captures and annotate deviations from steady-state for the various signal traces. These annotations will only be used to establish a baseline for the time required to extract events from time series data and will be used for objective validation metrics.

To annotate the files, open the program in your analytic environment. Record your start time then open the data file and apply the provided formatting script. Next, you will use the 'event' feature in the program to annotate the start and end of deviations from steady-state in each of the data traces. Treat these markers as binary on/off (or start/stop) for events placed in the given signal trace. For example, an engine turn-on event would likely include brake pressure and a change in the engine RPMs. Separately, add annotations that mark the brake press and release in the brake channel and mark the start of the engine RPM increase and end once the RPMs settle to idle. You are looking for changes in the data associated these types of events:

- Engine turn-on
- Left and Right turns
- Left and Right lane changes
- Complete stops and acceleration from complete stops
- Changes in Transmission State

Please ensure that 'start' event markers have an 'end' event marker unless the event is carried through the end of the file.

Lastly, record your end time and save the events file using the appropriate filename format. Repeat these steps for all six of your assigned files.
You may take any number of breaks you require during your survey or annotation time. Please include notations of the beginning and end of the breaks to best capture an overall time required to analyze each file.

Part 2) Machine-Learning Method—----------
You will then be provided an automation and a second set of six (6) collected CAN logs. You will be using the autonomous event extraction technique and marking the same candidate events as in B.

For each file, please record your start time. Then run the automation. Review the outputted candidate events and make any corrections you find. Record the time at which you complete. Repeat this process for all six assigned files.

The final step will be to complete a post-survey. This survey is about your experience and opinions about the event-extraction automation.

Direction Summary
Part 1—----------

1. Take pre-survey
2. Annotate assigned files
    1. Open event logging program
    2. Record start time
    3. Follow steps to format data
    4. In each of the data traces, add on/off 'event' markers
    5. Record end time
    6. Calculate time required to extract data less break time (rounded to minute)
    7. Save event markers file as 'filenumber_lastname_XXmin.xml' (e.g. 07_wightman_14min.xml)
    8. Repeat for other assigned files

Part 2—----------

3. Run autonomous event extraction for assigned file and observe outputs
    1. Record start time
    2. Use pre-trained models associated with Step 2 assigned files
    3. Run event extraction program for one (1) assigned file
    4. Validate extracted events
    5. Record end time
    6. Calculate the time required to run the program and validate the output less break times (rounded to minute)
    7. Repeat for other assigned files
4. Complete post-survey

Appendix A.2 – Pre-Survey

Pre-Survey

1. Respondent information

1.1. How many years of experience do you have analyzing time series data?

<1	<5	<10	<15	15+
1	2	3	4	5

1.2. How confident are you in your ability to analyze time series data?

Not Confident	Minimally Confident	Somewhat Confident	Confident	Very Confident
1	2	3	4	5

1.3. How familiar are you with artificial intelligence, machine-learning, or deep-learning concepts/research?

Very Unfamiliar	Unfamiliar	Somewhat Familiar	Familiar	Very Familiar
1	2	3	4	5

1.4. How important do you believe autonomous event extraction could be for your application domain?

Not Important	Minimally Important	Somewhat Important	Important	Very Important
1	2	3	4	5

1.5. How confident are you that machine-learning methods can accelerate your analytic workflow for event extraction?

Very Skeptical	Skeptical	Indifferent	Confident	Very Confident
1	2	3	4	5

2. Current analytic methods

2.1. How difficult would you characterize man-in-the-loop event extraction?

Not Difficult	Minimally Difficult	Somewhat Difficult	Difficult	Very Difficult
1	2	3	4	5

2.2. How long do you estimate it takes to manually extract events from a typical formatted, but new data file for a system you are familiar with?

<5 mins	<15 mins	<30 mins	<45 mins	>1 hr
1	2	3	4	5

2.3. How long do you estimate it takes to manually extract events from a typical formatted, but new data file for a system you are unfamiliar with?

<5 mins	<15 mins	<30 mins	<45 mins	>1 hr
1	2	3	4	5

2.4. What percentage, during an average week, do you estimate you spend extracting events from formatted data?

<5%	<15%	<30%	<45%	>60%
1	2	3	4	5

2.5. How confident are you that your current process methods catches all relevant “events”?

Very Skeptical	Skeptical	Indifferent	Confident	Very Confident
1	2	3	4	5

2.6. How timely would you say your current methods are? i.e. Do your current methods output analysis deliverables (e.g. reports or models) at the speed required by customers?

Not Timely	Minimally Timely	Somewhat Timely	Timely	Very Timely
1	2	3	4	5

3. Initial perceptions

3.1. How difficult would you characterize autonomous (computer-based) event extraction?

Not Difficult	Minimally Difficult	Somewhat Difficult	Difficult	Very Difficult
1	2	3	4	5

3.2. How much time savings do you believe future autonomous extraction methods could save during an average week when considering the required validation of automated outputs?

No Savings or More Work	**Minimal Savings (<1% effort)**	**Some Savings (<5% effort)**	**Decent Savings (<10% effort)**	**Significant Savings (>10% effort)**
1	2	3	4	5

3.3. Describe your experiences with past efforts at automating event extraction automations. (Short Answer)

3.4. Describe your concerns with automated event extraction methods. (Short Answer)

Appendix A.3 – Post-Survey

Post-Survey

1. Method Interactions

1.1. How would you characterize the ease of the event detection method?

Very Difficult	Difficult	Indifferent	Easy	Very Easy
1	2	3	4	5

1.2. Characterize your approximate time and effort savings factoring in both program run-time and output validation.

No Savings or More Work	Minimal Savings (<1% effort)	Some Savings (<5% effort)	Decent Savings (<10% effort)	Significant Savings (>10% effort)
1	2	3	4	5

1.3. How did your thinking about automated event extraction change after running this method? (Short Answer)

2. Method perceptions

2.1. How confident are you in your validation assessment of the autonomously extracted events?

Not Confident	Minimally Confident	Somewhat Confident	Confident	Very Confident
1	2	3	4	5

2.2. How useful do you believe this autonomous event extraction method could be for your application domain?

Not Useful	Minimally Useful	Somewhat Useful	Useful	Very Useful
1	2	3	4	5

2.3. How would you characterize the trustworthiness of the event outputs for this CAN domain application? (i.e. Would you allow this program to present you with candidate events without re-validation?)

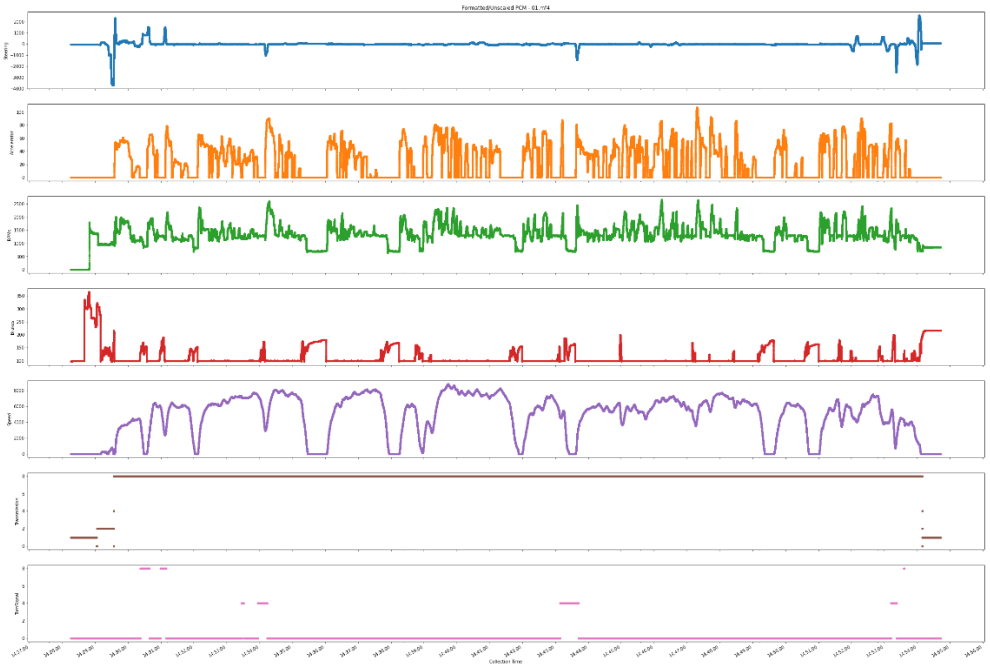
Not Trustworthy	Minimally Trustworthy	Somewhat Trustworthy	Trustworthy	Very Trustworthy
1	2	3	4	5

2.4. What factors increased your trust? (Short Answer

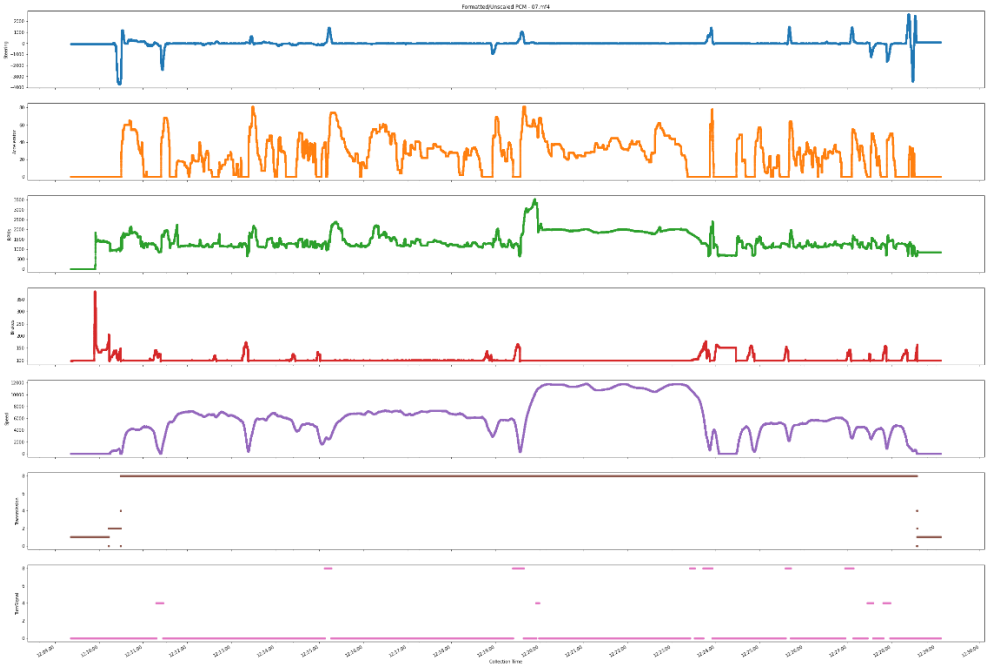
2.5. What factors decreased your trust? (Short Answer)

Appendix B – Sample Controller Area Network Collections

File 01

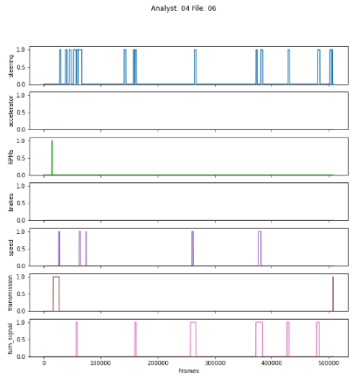


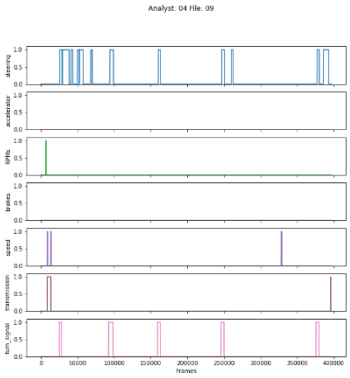
File 07



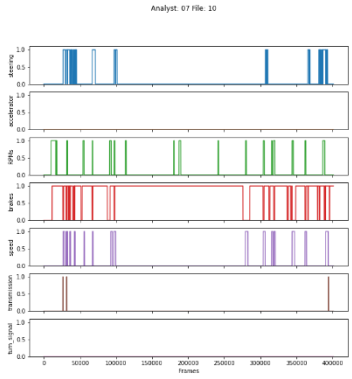
Appendix C – Sample Time Series Annotations

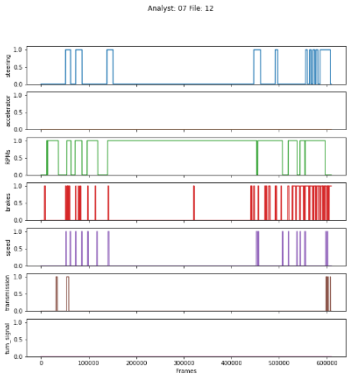
Analyst 04



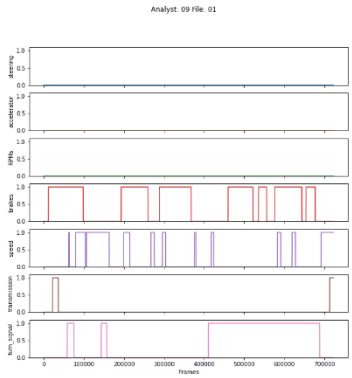


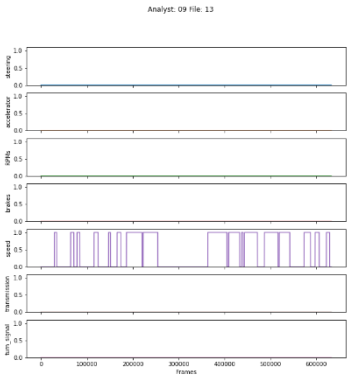
Analyst 07





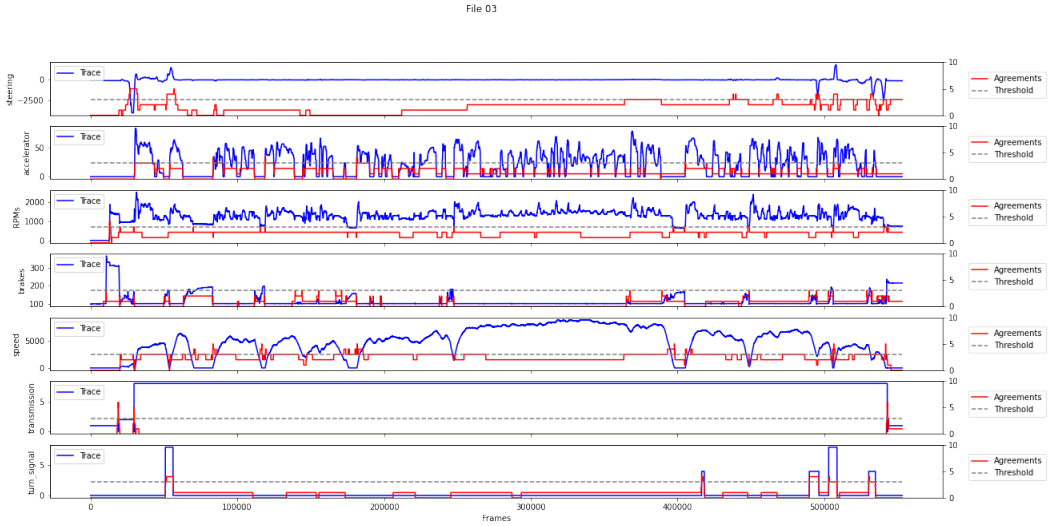
Analyst 09



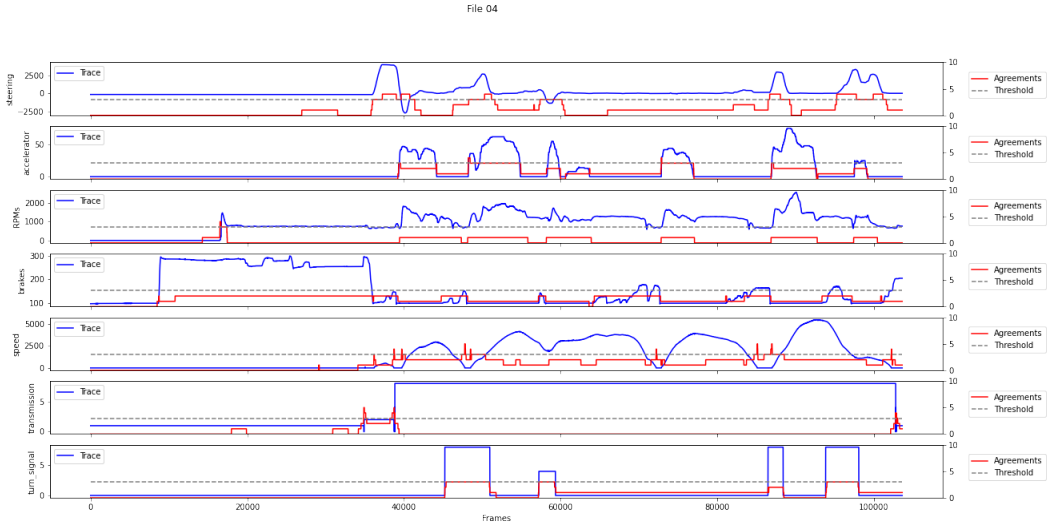


Appendix D – Combined Expert Annotations

File 03



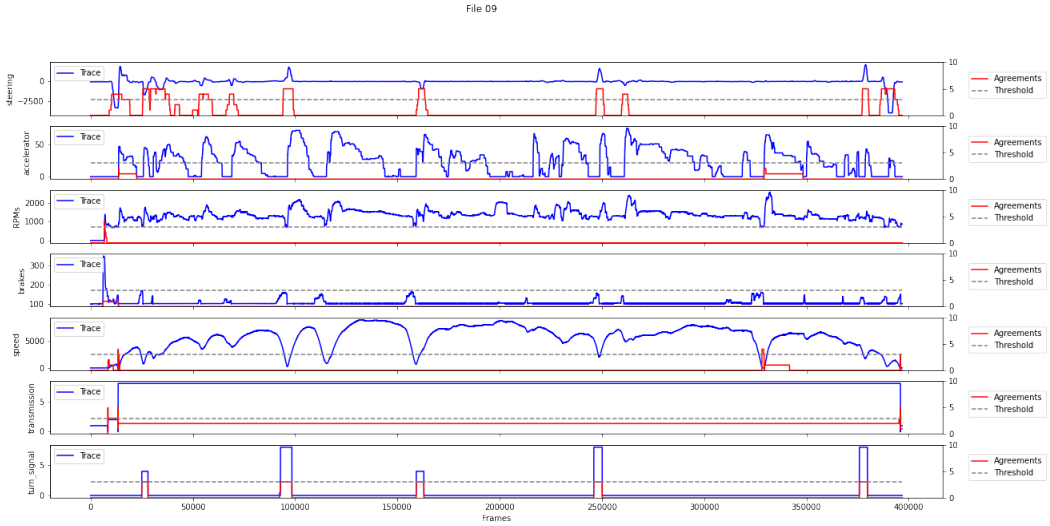
File 04



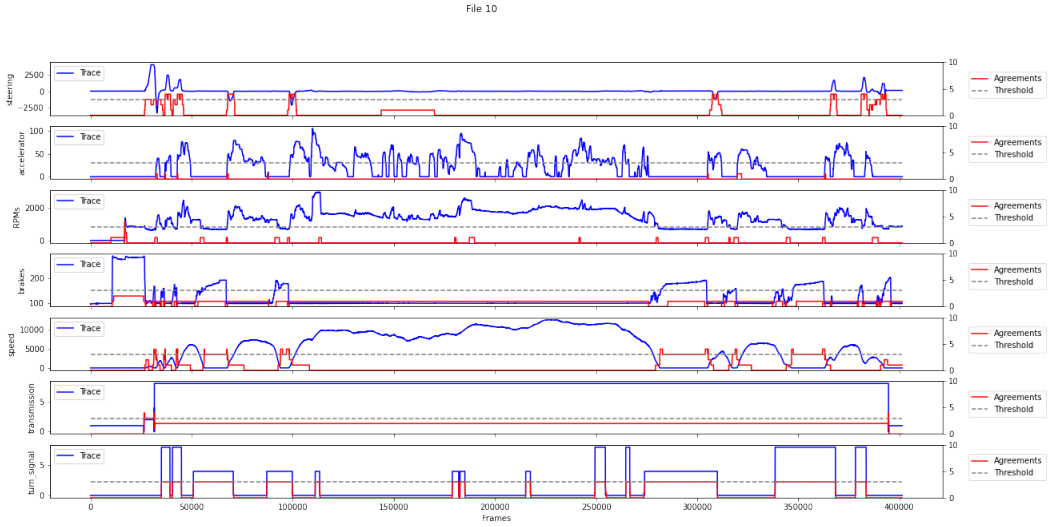
File 05

File 06

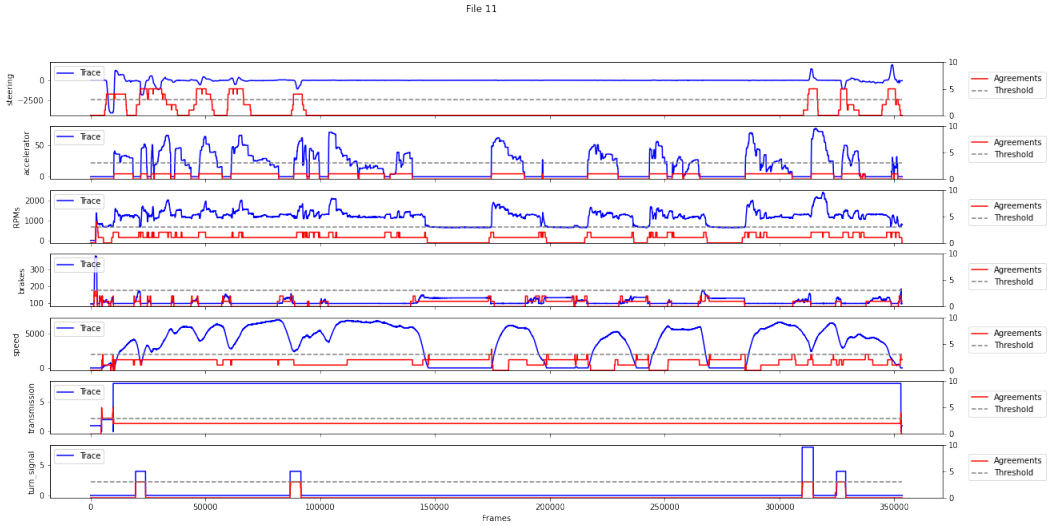
File 09



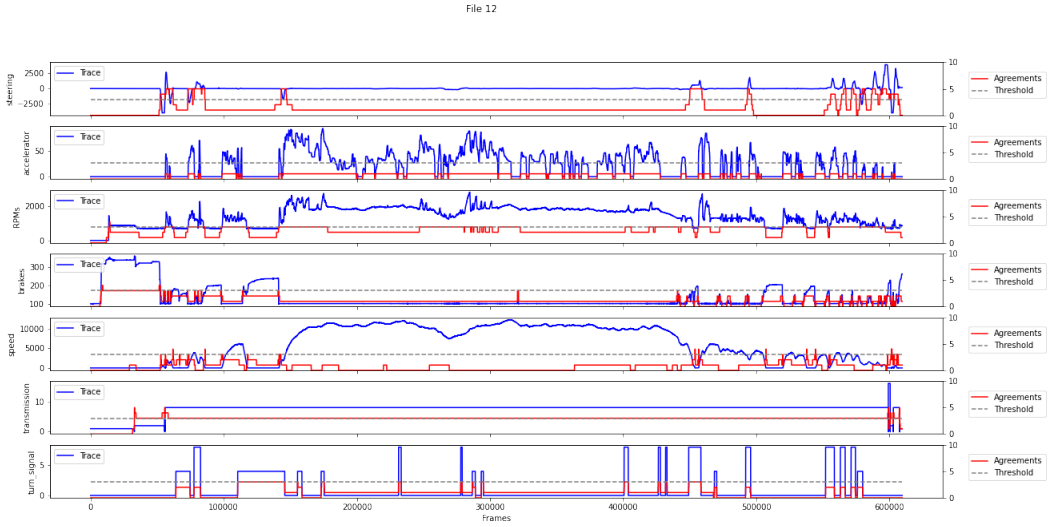
File 10



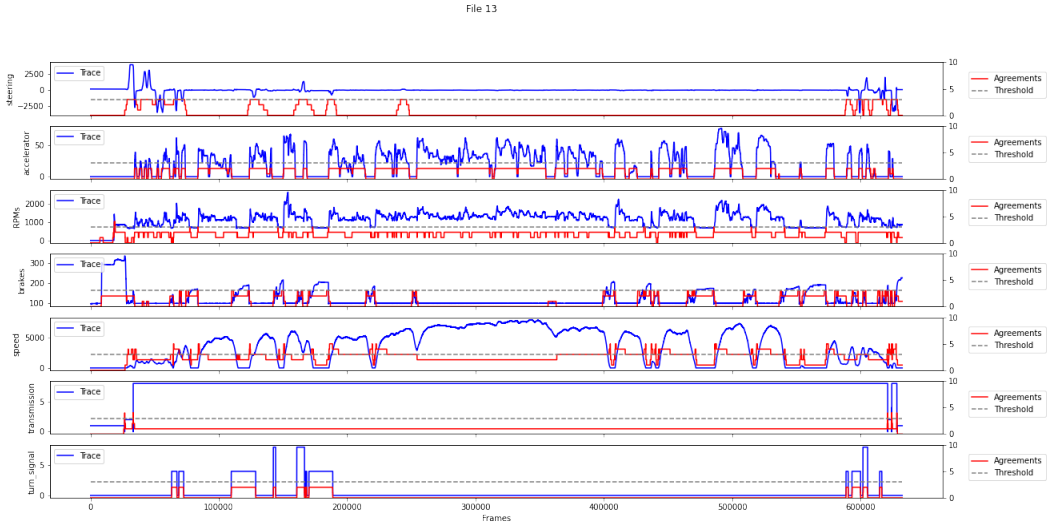
File 11



File 12



File 13


File 13
steering
accelerator
RPMs
brakes
speed
transmission
turn_signal
Trace
Agreements
Threshold
Frames
0
100000
200000
300000
400000
500000
600000

Bibliography

[1] J. Gen. Charles Q. Brown, "Accelerate Change or Lose," 31 August 2020. [Online]. Available: https://www.af.mil/Portals/1/documents/csaf/CSAF_22/CSAF_22_Strategic_Approach_Accelerate_Change_or_Lose_31_Aug_2020.pdf. [Accessed 23 November 2021].

[2] J. S. Knapp, "Facilitating Automated Machine to Machine Protocol Analysis," *Theses and Dissertations,* Vols. AFIT-ENG-DS-20-S-094, 2019.

[3] B. C. Stone, "Enabling Auditing and Intrusion Detection of Proprietary Controller Area Networks," 2018. [Online]. Available: https://scholar.afit.edu/etd/1940.

[4] National Science Foundation, "Cyber-Physical Systems: Enabling a Smart and Connected World," National Science Foundation, [Online]. Available: https://www.nsf.gov/news/special_reports/cyber-physical/. [Accessed 13 December 2021].

[5] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield and M. D. Iannacone, "CAN-D: A Modular Four-Step Pipeline for Comprehensively Decoding Controller Area Network Data," *IEEE Transactions on Vehicular Technology,* vol. 10, no. 1109, 2021.

[6] K. Hundman, V. &. L. C. Constantinou, I. Colwell and T. Soderstrom, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *24th ACM SIGKDD International Conference Authors*, 2018.

[7] W.-H. Lee, J. Ortiz, B. Ko and R. Lee, "Time Series Segmentation through Automatic Feature Learning," *arXiv,* vol. 1801, no. 05394, 2018.

[8] H. Meng, Y. Zhang, Y. Li and H. Zhao, "Spacecraft anomaly detection via transformer reconstruction error," University of Toronto Institute of Aerospace Studies, 07 2019. [Online]. Available: http://www.utias.utoronto.ca/wp-content/uploads/2019/07/88-Spacecraft-anomaly-detection-via-transformer-reconstruction-error.pdf. [Accessed 26 10 2021].

[9] M. Braei and S. Wagner, "Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art," *arXiv:2004.00433v1,* 2022.

[10] A. Mahmoud and A. Mohammed, "A Survey on Deep Learning for Time-Series Forecasting," in *Machine Learning and Big Data AnalyticsParadigms: Analysis, Applications and Challenges, Studies in Big Data 77*, Springer Nature Switzerland, Springer, 2021, pp. 365-392.

[11] M. Goldstein and S. Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PLoS ONE,* vol. 4, no. 11, 2016.

[12] S. Aminikhanghahi and D. J. Cook, "A Survey of Methods for Time Series Change Point Detection," *Knowl Inf Syst.,* vol. 51, no. 2, 2017.

[13] C. Truong, L. Oudre and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing,* vol. 167, no. 107299, 2020.

[14] S. Makridakis, E. Spiliotis and a. V. Assimakopoulos, "Statistical and machine learningforecasting methods: Concerns and ways forward," *PLOS ONE,* vol. 13, no. 3, pp. 1-26, 2018.

[15] G. Aurelien, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, Sebastopol, CA: O'Reilly Media, Inc, 2019.

[16] Philosiblog, "If one does not know to which port one is sailing, no wind is favorable.," Philosiblog, 30 July 2011. [Online]. Available: https://philosiblog.com/2011/07/30/if-one-does-not-know-to-which-port-one-is-sailing-no-wind-is-favorable/. [Accessed 5 August 2022].

[17] C. C. Aggarwal, Outlier Analysis, vol. 2nd, Springer Publishing Company, Incorporated, 2016.

[18] G. E. P. Box and G. Jenkins, Time Series Analysis, Forecasting and Control, San Francisco, CA, USA: Holden-Day Inc., 1990.

[19] R. J. Hyndman, A. B. Koehler, R. D. Synder and S. Grose, " state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting,* vol. 18, no. 3, pp. 439-454, 2002.

[20] Y. Yu, Y. Zhu, S. Li and D. Wan, "Time series outlier detection based on sliding window prediction," *Mathematical Problems in Engineering,* 2014.

[21] M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, "DeepANT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," *IEEE Access,* vol. 7, pp. 1991-2005, 2019.

[22] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation," in *CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE*, 1985.

[23] P. Elspas, Y. Klose, S. Isele, J. Bach and E. Sax, "Time Series Segmentation for Driving Scenario Detection with Fully Convolutional Networks," in *7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, 2021.

[24] S. Feike, "Multiple Time Series Classification by Using Continuous Wavelet Transformation," Towards Data Science, 1 February 2020. [Online]. Available: https://towardsdatascience.com/multiple-time-series-classification-by-using-continuous-wavelet-transformation-d29df97c0442. [Accessed 5 August 2022].

[25] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, A. Graves, N. Kalchbrenner, A. Senior and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv,* 2016.

[26] M. Perslev, M. H. Jensen, S. Darkner, P. J. Jennum and C. Igel, "U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging," vol. arXiv:1910.11162v1, 2019.

[27] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR,* vol. arXiv:1512.03385, 2015.

[28] H. Lai, H. Chen and S. Wu, "Different Contextual Window Sizes Based RNNs for Multimodal Emotion Detection in Interactive Conversations," *IEEE Access,* vol. 8, pp. 119516-119526, 2022.

[29] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation.," *arXiv:1406.1078,* 2014.

[30] S. Chowdary, "Generating Your Shakespeare Text Using Sequential Models Such As Long-Short-Term-Memory (LSTMs), Gated Recurrent Units (GRUs), Recurrent Neural Network (RNNs)," MARKTECHPOST, 1 April 2021. [Online]. Available: https://www.marktechpost.com/2021/04/01/generating-your-shakespeare-text-using-sequential-models-such-as-long-short-term-memory-lstms-gated-recurrent-units-grus-recurrent-neural-network-rnns/. [Accessed 4 August 2022].

[31] S. Flores, "Variational Autoencoders are Beautiful," Comp Three Inc., 15 April 2019. [Online]. Available: https://www.compthree.com/blog/autoencoder/. [Accessed 4 August 2022].

[32] S. Cristina, "The Attention Mechanism from Scratch," Machine Learning Mastery, 20 September 2021. [Online]. Available: https://machinelearningmastery.com/the-attention-mechanism-from-scratch/. [Accessed 4 August 2022].

[33] S. Serrano and N. A. Smith, "Is Attention Interpretable," in *57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 2017.

[35] O. Press, N. A. Smith and M. Lewis, "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation," vol. arXiv:2108.12409v1, 2021.

[36] E. Aksan, M. Kaufmann, P. Cao and O. Hilliges, "A Spatio-temporal Transformer for 3D Human Motion Prediction," vol. arXiv:2004.08692v3, 2021.

[37] J. Ma, Z. Shou, A. Zareian, H. Mansour, A. Vetro and S.-F. Chang, "CDSA: Cross-Dimensional Self-Attention forMultivariate, Geo-tagged Time Series Imputation," vol. arXiv:1905.09904v2.

[38] D. Neimark, O. Bar, M. Zohar and D. Asselmann, "Video Transformer Network," vol. arXiv:2102.00719v3, 2021.

[39] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart and M. Brubaker, "Time2Vec: Learning a Vector Representation of Time," *Borealis AI,* vol. arXiv:1907.05321v1, 2019.

[40] R. J. Hyndman and G. Athanasopoulos, "6.6 STL decomposition," in *Forecasting: Principles and Practice*, Monash University, Australia, OTexts.com/fpp2, 2018.

[41] M. Goldstein and A. Denegel, "Hisogram-based Outlier Score: A fast Unsupervised Anomaly Detection Algorithm," in *KI-2012: Poster and Demo Track*, 2012.

[42] B. M. M, H. P. Kriegel, R. T. Ng and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, 200.

[43] Z. He, X. Xu and S. Deng, "Discovering Cluster-based Local Outliers," *Pattern Recognition Letters,* vol. 24, no. 9-10, pp. 1641-1650, 2003.

[44] M. Goldstein, "Anomaly Detection in Large Datasets," [PhD-Thesis]. University of Kaiserslautern., München, Germany, 2014.

[45] P. C. Mahalanobis, "On the Generalized Distance in Statistics," in *Proceedings National Institute of Science*, India, 1936.

[46] R. Kwitt and U. Hofmann, "Unsupervised Anomaly Detection in Network Traffic by Means of Robust PCA," in *In:Proceedings of the International Multi-Conference on Computing in the Global Information Technology (ICCGI'07)*, Washington DC, USA, 2007.

[47] L. Shen, Z. Yu, Q. Ma and J. T. Kwok, "Time Series Anomaly Detection with Multiresolution Ensemble DecodingLifeng," in *TheThirty-FifthAAAIConferenceonArtificial Intelligence(AAAI-21)9567* , 2021.

[48] "Outlier Detection for Time Series with Recurrent Autoencoder EnsemblesTung," in *Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)2725 For*, 2019.

[49] K. H. Rosen, Discrete Mathematics and its Applications, Eighth Edition, New York, NY: McGraw-Hill Education, 2019.

[50] K. Fortney, "Machine Learning — An Error by Any Other Name…," Towards Data Science, 8 January 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-an-error-by-any-other-name-a7760a702c4d. [Accessed 4 August 2022].

[51] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer and C.-J. Hsieh, "Large Batch Optimization For Deep Learning: Training BERT in 76 Minutes," in *ICLR*, 2020.

[52] J. Bergstra, R. Bardenet, Y. Bengio and B. Kegl, "Algorithms for hyper-parameter optimization," 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf. [Accessed 4 August 2022].

[53] J. W. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 4th Edition, Thousand Oaks, CA: SAGE Publications, Inc., 2014.

[54] M. B. Miles and A. M. Huberman, Qualitative data analysis: A sourcebook of new methods, Thousand Oaks, CA: SAGE, 1994.

[55] C. Geisler and J. Swarts, Coding Streams of Language: Techniques for the Systematic Coding of Text, Talk, and Other Verbal Data, The WAC Clearinghouse; University Press of Colorado, 2019.

[56] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica,* vol. 22, no. 3, pp. 276-282, 2012.

[57] P. Zhang and Z. Obradovic, "Integration of Multiple Annotators by Aggregating Experts and Filtering Novices," in *2012 IEEE International Conference on Bioinformatics and Biomedicine*, 2012.

[58] S. Paun and E. Simpson, "Aggregating and Learning from Multiple Annotators," in *EACL: Tutorials*, 2020.

[59] "Controller Area Network (CAN) Overview," 1 September 2020. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/06/controller-area-network--can--overview.html. [Accessed 5 August 2022].

[60] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak and I. Sutskever, "Deep Double Descent: Where Bigger Models and More Data Hurt," vol. arXiv:1912.02292v1, 2019.

[61] M. B.́kowski, G. Marti and P. Donnat, "Autoregressive Convolutional Neural Networks for Asynchronous Time Series," vol. arXiv:1703.04122v4, 2018.

[62] M. C. McCallum, "Unsupervised Learning of Deep Features for Music Segmentation," *ICASSP,* pp. 346-350, 2019.

[63] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu and J. Engel, "Encoding Musical Style with Transformer Autoencoders," in *37th International Conference on Machine Learning*, Vienna, Austria, 2020.

[64] M. Buccoli, M. Zanoni, A. Sarti, S. Tubaro and D. Andreoletti, "Unsupervised feature learning for Music Structural Analysis," in *24th European Signal Processing Conference (EUSIPCO)*, 2016.

[65] P. Schäfer, A. Ermshaus and U. Leser, "ClaSP - Time Series Segmentation," in *International Conference on Information and Knowledge Management*, 2021.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
06-08-2022	Master's Thesis	October 2020 – August 2022

TITLE AND SUBTITLE

Analytic Case Study Using Unsupervised Event Detection in Multivariate Time Series Data

5a. CONTRACT NUMBER

5b. GRANT NUMBER
F4FTBJ0287JW01

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Wightman, Jeremy M., Civilian, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENP-MS-22-S-051

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Air and Space Intelligence Center
4180 Watson Way
Wright-Patterson AFB, OH 45433

10. SPONSOR/MONITOR'S ACRONYM(S)

NASIC

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

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Analysis of cyber-physical systems (CPS) has emerged as a critical domain for providing US Air Force and Space Force leadership decision advantage in air, space, and cyberspace. Legacy methods have been outpaced by evolving battlespaces and global peer-level challengers. Automation provides one way to decrease the time that analysis currently takes. This thesis presents an event detection automation system (EDAS) which utilizes deep learning models, distance metrics, and static thresholding to detect events. The EDAS automation is evaluated with case study of CPS domain experts in two parts. Part 1 uses the current methods for CPS analysis with a qualitative pre-survey and tasks participants, in their natural setting to annotate events. Part 2 asks participants to perform annotation with the assistance of EDAS's pre-annotations. Results from Part 1 and Part 2 exhibit low inter-coder agreement for both human-derived and automation-assisted event annotations. Qualitative analysis of survey results showed low trust and confidence in the event detection automation. One correlation or interpretation to the low confidence is that the low inter-coder agreement means that the humans do not share the same idea of what an annotation product should be.

15. SUBJECT TERMS
Cyber-physical systems, controller area network, event detection, time series analysis, case study

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gilbert Peterson, AFIT/ENG
U	U	U	UU	210	19b. TELEPHONE NUMBER *(Include area code)* 937-255-3636 x4281 (gilbert.peterson@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18