DISSERTATION

SPATIOTEMPORAL ANOMALY DETECTION:

STREAMING ARCHITECTURE AND ALGORITHMS

Submitted by

Barry S. Siegel

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2020

Doctoral Committee:

Advisor: John Labadie

Edwin Chong
Anthony Maciejewski
Peter Young

ABSTRACT

SPATIOTEMPORAL ANOMALY DETECTION: STREAMING ARCHITECTURE AND ALGORITHMS

Anomaly detection is the science of identifying one or more rare or unexplainable samples or events in a dataset or data stream. The field of anomaly detection has been extensively studied by mathematicians, statisticians, economists, engineers, and computer scientists. One open research question remains the design of distributed cloud-based architectures and algorithms that can accurately identify anomalies in previously unseen, unlabeled streaming, multivariate spatiotemporal data. With streaming data, time is of the essence, and insights are perishable. Real-world streaming spatiotemporal data originate from many sources, including mobile phones, supervisory control and data acquisition enabled (SCADA) devices, the internet-of-things (IoT), distributed sensor networks, and social media.

Baseline experiments are performed on four (4) non-streaming, static anomaly detection multivariate datasets using unsupervised offline traditional machine learning (TML), and unsupervised neural network techniques. Multiple architectures, including autoencoders, generative adversarial networks, convolutional networks, and recurrent networks, are adapted for experimentation. Extensive experimentation demonstrates that neural networks produce superior detection accuracy over TML techniques. These same neural network architectures can be extended to process unlabeled spatiotemporal streaming using online learning. Space and time relationships are further exploited to provide additional insights and increased anomaly detection accuracy.

A novel domain-independent architecture and set of algorithms called the **S**patio**t**emporal **A**nomaly **D**etection **E**nvironment (STADE) is formulated. STADE is based on federated learning architecture. STADE streaming algorithms are based on a geographically unique, persistently executing neural networks using online stochastic gradient descent (SGD). STADE is designed to be pluggable, meaning that alternative algorithms may be substituted or combined to form an ensemble. STADE incorporates a Stream Anomaly Detector (SAD) and a Federated Anomaly Detector (FAD). The SAD executes at multiple locations on streaming data, while the FAD executes at a single server and identifies global patterns and relationships among the site anomalies. Each STADE site streams anomaly scores to the centralized FAD server for further spatiotemporal dependency analysis and logging. The FAD is based on recent advances in DNN-based federated learning.

A STADE testbed is implemented to facilitate globally distributed experimentation using low-cost, commercial cloud infrastructure provided by Microsoft™. STADE testbed sites are situated in the cloud within each continent: Africa, Asia, Australia, Europe, North America, and South America. Communication occurs over the commercial internet. Three STADE case studies are investigated. The first case study processes commercial air traffic flows, the second case study processes global earthquake measurements, and the third case study processes social media (i.e., Twitter™) feeds. These case studies confirm that STADE is a viable architecture for the near real-time identification of anomalies in streaming data originating from (possibly) computationally disadvantaged, geographically dispersed sites. Moreover, the addition of the FAD provides enhanced anomaly detection capability. Since STADE is domain-independent, these findings can be easily extended to additional application domains and use cases.

ACKNOWLEDGEMENTS

I want to thank the entire College of Engineering faculty and staff for supporting this dissertation. I would also particularly like to thank the committee chair, Professor John Labadie, for the support provided over many years, the committee members, Professor Chong, Professor Maciejewski, and Professor Young for their constructive feedback. Professor Labadie's course on optimization and operations research was the genesis of this dissertation. I would also like to thank my entire Ph.D. committee for the informative System Engineering courses, which provided the scientific basis of this dissertation. Finally, I would like to thank Ingrid Bridge from the Department of Systems Engineering, who provided amazing guidance traversing the requirements of the System Engineering Ph.D. program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Background

Anomaly detection is the science of identifying novelties, non-conforming patterns in data [1]. Anomalies are known in the literature as noise, deviations, and exceptions. The study of anomaly detection has a long history in multiple disciplines, including engineering, statistics, economics, bioinformatics, and geoinformatics. Recent research has been focused on the financial, cyber, robotics, and medical application domains. Fraud detection [2], intrusion detection [3] and [4], data center management [5], [6] and [7], financial markets [8], robotics [9], smart buildings [10], petroleum industry applications [11], computer network traffic [12], software verification [13], water treatment [14], [15] and [16], and wireless networks [17] and [18] are popular domains that have been addressed by anomaly detection architectures and algorithms.

A variety of traditional machine learning (TML) algorithms has been applied to anomaly detection problems. These algorithms range from stochastic models, time-series models, and classification, clustering, and nearest neighbor non-parametric techniques. Most of the approaches have been designed for small datasets and are non-scalable. Few studies have addressed the requirement for large-scale, multivariate, streaming anomaly detection.

There has been a well-advertised renaissance in the field of artificial intelligence and the use of deep neural networks (DNN) to address complex engineering problems such as computer vision, natural language processing, and robotics. While research has progressed on the application of DNNs to large-scale anomaly detection [19], progress has been hampered by the lack of quality datasets [20]. Anomaly detection experimentation has resorted to the production of synthetically generated labeled data to conduct experiments. Perhaps more significantly, the

greater focus now has been placed on the development of unsupervised or self-supervised learning techniques to exploit a large number of unlabeled datasets.

Why DNNs for anomaly detection? The short answer is that TML algorithms have proven to be insufficiently robust for application to complex anomaly detection problem domains. TML algorithms produce anomaly scores on individual sample points, which is also known as point anomalies. These scores are often interpreted as an anomaly probability. TML algorithms do not transfer well to problem domains with streaming requirements, difficult to decipher spatial inter-relationships, long time horizons, and complex multivariate interactions. TML algorithms also produce mediocre results because of the core underlying model assumptions of stationary data and processes are often erroneous. In the presence of non-stationary data and processes, TML-based anomaly detection techniques may produce suboptimal anomaly classifications.

DNNs are particularly suited for the estimation of complex, data-driven behaviors when the underlying generating model is uncertain or unknown [21]. Recent advances in neural machine translation may be used as a template for potential advances in anomaly detection algorithms [22]. For example, a language translation model can be formulated as a sequence-to-sequence prediction model with compressed representations of sentences encoded for efficiency. Analogously, a spatiotemporal anomaly detection problem may also be formulated as a temporal-to-temporal prediction problem with encoded data representations; anomalies are identified when the predicted representation of the sequence of events deviates substantially for the observed representation.

1.2 Engineering Challenge

The growth of the internet has created an explosion of streaming data generated from social networking web sites, mobile devices, and the internet-of-things (IoT). Supervisory control and data acquisition (SCADA) software and hardware components that monitor and process real-time data at geographically distributed sites create massive volumes of streaming data. Ubiquitous embedded devices such as space-based sensors, robotics, and autonomous self-driving vehicles also generate raw stream data. In these cases, robust, near real-time anomaly detection techniques are required for optimal functionality and safety.

The engineering challenge is to design domain-independent architectures and algorithms that exploit the underlying or hidden spatial and temporal (spatiotemporal) relationships that contribute to the timely identification of anomalies in multivariate data streams. The definition of timely is domain-dependent or analyst-defined. The term spatiotemporal can be broadened to include sequential data without a time dimension (e.g., a DNA sequence). The terms temporal and sequential are used interchangeably, but the focus here is on time and geospatially dependent streams.

1.3 Organization

Chapter 1 (this chapter) provides a general overview; spatiotemporal anomaly detection is placed into perspective by presenting several real-world use cases. Chapter 2 highlights some foundational topics, including the types of anomalies, anomaly labeling and scoring, the use of unsupervised techniques, and anomaly detection performance evaluation. Chapter 3 presents the four (4) datasets used for TML and DNN experimentation, while Chapter 4 discusses the nine (9) different TML techniques used for experimentation. The discussion also includes a literature review. Chapter 5 then discusses the TML experimentation results. The purpose of Chapters 4

and 5 is not to provide a comprehensive evaluation of TMLs as applied to anomaly detection problems; instead, the discussion provides a sharp contrast to the DNNs presented in Chapter 6. Various flavors of autoencoders, generative adversarial networks, and recurrent neural networks are discussed in Chapter 6; these representational learning models, also known in the popular science literature as deep learning, are a vital ingredient to the **S**patio**t**emporal **A**nomaly **D**etection **E**nvironment (STADE). Chapter 6 also provides an intensive literature review of recent applications of representational learning to anomaly detection, most of which have appeared in the last two to three years. Chapter 7 provides experimentation results associated with the DNNs using the same datasets described in Chapter 3. These results are compared and contrasted with the TML results to illustrate the performance benefits of DNNs. Leveraging the recent advances in commercial cloud computing, Chapter 8 then presents the STADE specification and testbed for distributed real-time streaming spatiotemporal data, including a Stream Anomaly Detector (SAD) and a Federated Anomaly Detector (FAD). The purpose of the FAD is to globally accumulate SAD scores and provide feedback back to the SAD geographically distributed sites. Within STADE specification, there is an architecture component, an algorithm component, and a testbed component. The architecture component addresses how geospatially distributed sites orchestrate and communicate with each other. The algorithm component addresses anomaly detection issues, neural networks, computation speed, accuracy, and other topics of importance to DNN stream processing. The STADE testbed is a cloud-based instantiation of the STADE architecture used for experimentation. Chapter 9 describes experimentation with the STADE testbed, including the data sources, estimation, and the results of the global air-traffic case study; Chapter 10 describes the experimentation with the global earthquake case study, and Chapter 11 describes experimentation with the social

networking (i.e., Twitter™) case study. The first two case studies are engineering and earth-science related, while the third case study is text related. Social media is viewed as a network of human sensors, so the distinction is not critical; the architecture and algorithms described are entirely domain independent. Finally, Chapter 12 provides a summary of the results, research limitations, and recommendations for future research.

1.4 Use Cases

Use cases are a technique in systems engineering to explore essential concepts of an underlying architecture or algorithm designed to solve real-world problems. Use cases can be specified formally using modeling languages (e.g., Unified Modeling Language) or informally using textual descriptions or flow diagrams. Examples of streaming anomaly detection use cases include (1) Air Traffic Control (ATC): automatic tracking of flight anomalies and deviations from normal operations; (2) Connected and Autonomous Vehicle (CAV): runtime identification of hazards supporting the autonomous operation of vehicles; (3) Cyber-Physical System (CPS): the recognition of cyber issues such as coordinated denial-of-service, network intrusion, and other attacks in CPSs such as smart buildings and cloud data centers; (4) Distributed Sensor Network (DSN): timely fault detection of geospatially distributed sensors; (5) Earth Science (ES): the early warning of earthquakes, flooding, weather, and atmospheric anomalies; (6) Global Contraband (GC): the identification of anomalous global cargo shipping patterns and manifests designed to smuggle weapons of mass destruction and illicit drugs; (7) Global Pandemic (GP): the identification of anomalous global transmittal of viruses and associated infection and death rates; (8) Industrial Control (IC): the real-time identification of anomalies in SCADA hardware and software elements that monitor and control industrial devices; and (9)

Social Networking Stream (SNS): identification of anomalous events, changes in objectivity, sentiment, fake news, and bot attacks through the mining of social network data and text streams.

Table 1 summarizes the characteristics of these selected use cases in terms of the type of streaming data and the periodicity or frequency of the data. The required periodicity is dependent on the specifics of the real-world application. With these use cases, there is a temporal and spatial component that could be further exploited for spatiotemporal anomaly detection.

Table 1: Anomaly Detection Use Cases

Use Case	Streaming	Periodicity
Air Traffic Control (ATC)	Airplane Tracking	Minutes
Connected/Autonomous Vehicle (CAV)	Vehicle Traffic	Milliseconds
Cyber-Physical System (CPS)	Cyber	Seconds
Distributed Sensor Network (DSN)	Satellite Sensors	Milliseconds
Earth Science (ES)	Land Sensors	Seconds
Global Contraband (GC)	Cargo Shipping	Days/Weeks
Global Pandemic (GP)	Infections	Days
Industrial Control (IC)	Robotic Sensors	Seconds
Social Networking Stream (SNS)	Sentiment, Bots	Seconds

1.4.1 Air Traffic Control (ATC)

Air Traffic Control consists of a complex integrated system of pilots, controllers on the ground, radar ground stations, control towers, and software. Because of the volume of flights, flight anomalies may be undetected, resulting in catastrophic events. Actionable alerts may go unnoticed. Recent examples of flight-path anomalies and catastrophic failures that were not detected include Germanwings Flight 9525 [23] and Malaysia Airlines Flight 370 [24]. These failures were believed to be caused by intentional pilot behaviors. In both cases, had the anomalous flight path been detected in real-time, tragedies might have been prevented through early recognition through automated algorithms and the electronic intervention. Global Air Traffic (GAT) is the subject of the case study presented in Chapter 9.

1.4.2 Connected and Autonomous Vehicle (CAV)

CAVs rely on their local sensors and information received from other nearby vehicles and road structures to navigate the roadway safely. CAVs use wireless and near-field technologies to communicate. Hazard identification is critical to the operation and commercialization of CAVs. By their nature, CAVs generate spatiotemporal data. Streaming anomalous data generated through faulty sensors or a malicious local cyberattack could result in severe consequences, including fatal car crashes. The automated and timely detection of anomalies in real-time is critical to the long-term commercial success of CAVs.

1.4.3 Cyber-Physical System (CPS)

Foreign network attacks, host-based intrusions, malware, and other malicious cyber-attacks on CPSs [25], including the electrical grid, power plants, and water distribution networks, can have severe economic and security consequences. These attacks are generally recognized through the analysis of network packet attributes (e.g., the source and destination of the packet's Internet Protocol (IP) address) or the evaluation of terabyte-sized, text-based log files. Anomaly detection based on discrete event temporal sequences in log files is critical [26]; unfortunately, the recognition of these attacks is often too late to be actionable. Anomaly detection techniques that could recognize cyber-attacks in real-time by analyzing CPS streaming network traffic and log files would increase cybersecurity. Since cyberattacks may originate globally, initiated by adversarial nation-states, cyber data streams have a strong geographic component that could be exploited by the spatiotemporal aware anomaly detection algorithm.

1.4.4 Distributed Sensor Network (DSN)

DSNs are characterized by a spatially distributed set of autonomous devices used to monitor and log physical or environmental conditions. Often, these devices are located at the

network edge at disadvantaged locations where power and computer resources are limited. Extreme values of sensor readings might indicate a sensor fault or point anomaly. The anomaly detection algorithm must be able to detect the difference between a faulty sensor and valid but previously unseen data.

Through continuous monitoring, the spatial and temporal characteristics of DSN outputs can be exploited by analyzing not only the sequence of sensor reads at a particular location, but also by the relationship between the sequences at two or more sensor locations. Detection of an adversary missile launch, for example, could be formulated as an anomaly detection problem where the spatiotemporal pattern of sensor outputs from one satellite is time-related to the spatiotemporal pattern of outputs from a related satellite. Data can be transmitted remotely to a centralized server, aggregated, fused, and then deployed to a time-critical decision support system. For example, space-based sensors attached to multiple geo-spatially distributed satellite monitor and detect adversary missile launches. Anomalous sensor data could be an indicator of a sensor failure or a real adversarial missile launch.

1.4.5 Earth Science (ES)

Anomaly detection and related techniques applied to ES spatiotemporal domains, including global warming simulation, earthquake prediction, ozone level detection [27], ocean surface temperature monitoring, hurricane modeling, and the early warning of flood events [28]. ES has perhaps the most durable spatial component of all of the use cases listed in Table 1. This domain is heavily multivariate; for example, land cover anomalies often proxy previously unrecognized climate change and geological activity. Streaming weather shape data and satellite images used in conjunction with other earth science sensors may be inputs into contextual anomaly detection algorithms. In the future, earthquake prediction based on ES temporal and

spatial anomalous variations might be identified through representational learning algorithms. ES is the subject of the case study found in Chapter 10.

1.4.6 Global Contraband (GC)

Maritime Domain Awareness is the term used to describe the process of monitoring commodity import patterns and global ship movements [29]. Terrorists and rogue nations use cargo shipping as the primary means to smuggle GC, including nuclear weapon materials and illicit drugs. Algorithms that identify anomalous shipping transactions, fraudulent manifests, fake companies, and previously unseen ship movement patterns could be a capable detector of international smuggling and attempts to avoid financial sanctions [30].

1.4.7 Global Pandemic (GP)

A global pandemic is the rapid spread of a disease or virus across one or more regions. The H1N1 swine flu pandemic of 2009 and the COVID-19 pandemic of 2020 are examples of epidemics that originate one country and spread globally over time, sometimes over a few days or weeks. The spread of the disease or virus can be modeled, in part, as a spatiotemporal anomaly detection problem similar to a computer network virus. Geographic and sequential anomalies can be identified when particular regions exhibit low or high pandemic infection rates. Anomaly detection can be used in forecasting models to guide better public-policy decision making (e.g., to avoid draconian countermeasures) and to evaluate the reliability of published infection and death rate data by controlled governments.

1.4.8 Industrial Control (IC)

The Association for Computing Machinery 2017 Distributed and Event-Based Systems Grand Challenge focused on the problem of the analysis of anomalies in streaming IC data generated by digital and analog sensors embedded within manufacturing equipment [31].

Factory floor equipment failures in the manufacturing process often result in defective products. Equipment failures may occur randomly or incrementally over time. Automated diagnosis of industrial equipment using anomaly detection techniques could result in the early identification of faulty product manufacturing and failing equipment. Delayed recognition of faulty manufacturing processes increases product rework expenses and other costs. Unusual shapes of temporal values originating from equipment sensors may indicate an emergent equipment failure or the need for preventative maintenance.

1.4.9 Social Networking Stream (SNS)

Social networks such as Twitter™, Instagram™, and YouTube™ generate massive geographically distributed SNSs and images with annotations. SNSs effectively form a grid of human sensors. These SNSs provide valuable, timely, and actionable intelligence and situational awareness on evolving current events, natural emergencies, adversarial cyberattacks, and consumer sentiment. The volume and variety of these SNSs create a technical barrier to their exploitation. Anomaly detection techniques can be used to identify critical social networking information that would otherwise be lost in the noise by traditional algorithms. SNSs are the subject of the case study presented in Chapter 11.

1.5 Research Contributions and Limitations

The contributions of this research are as follows:

1. A benchmark comparison between TML and DNN algorithms to identify various types of anomalies using datasets from multiple application domains.
2. A novel STADE architecture that employs a continuous running DNN with parameter updates in a unified framework to identify anomalies in multivariate, streaming spatiotemporal data. Within the STADE architecture is a Stream Anomaly Detector

(SAD) and a Federated Anomaly Detector (FAD) that identifies relationships among LAD anomalies.

3. Three cloud-based case study demonstrating architecture and algorithmic concepts of STADE using live commercial air traffic flow from globally positioned sensors, live global earthquake readings from global sensors, and streaming tweets from the social network provider Twitter™.

In order to focus on the algorithms associated with spatiotemporal anomaly detection, there are several related albeit essential topics that will not be addressed in detail. The massive growth in the accuracy and performance of representational learning has been a result, in part by the availability of powerful Graphical Processing Units from NVIDIA™, Tensor Processing Units from GOOGLE™, and various Application-Specific Integrated Circuits. NVIDIA's parallel computing infrastructure can speed-up representational learning calculations by orders of magnitude over sequentially-based programming. Optimization and performance enhancements through the use of parallel computing architectures are not explicitly addressed herein.

Seasonality in temporal data is typical but is not incorporated into the anomaly detection algorithms. Similarly, there are conceptual differences between locally distributed and globally distributed anomaly detection algorithms. Local distribution may execute within a data center or factory floor, while global distribution may be across the world in the commercial cloud. The focus of the case study is on the commercial cloud, but the results could be extended to local, clustered-based computing.

Image anomaly detection, an emerging application area in representational learning, is not addressed. For example, automatic machinery surface image inspection for fault detection, building entry image detection for anomalous human activity, automated analysis of temporal x-

rays images for medical diagnosis are examples of the application of anomaly detection algorithms. Convolutional neural networks (CNNs) is the neural network architecture primarily used in vision problems. While vision anomaly detection problems are not addressed, CNNs adapted to non-vision multivariate anomaly problems are discussed in Chapters 6 and 7.

Despite the explosion in the deployment of machine learning models for decision support, research in understanding the underlying reasoning behind representational learning is an open research problem. Explainable AI is essential, especially in mission-critical domains, because decision-makers need to have trust in a model override of personal expertise and intuition [32], [33]. Explaining anomaly designations produced by neural networks and trust versus untrustworthy models are critical topics that will not be addressed. For a recent comprehensive survey of techniques for explaining black-box models, see [34] and [35].

1.6 List of Acronyms

Table 2: List of Acronyms

1D-CNN	One Dimensional Convolutional Neural Network
ATC	Air Traffic Control
AUC	The area under the (ROC) Curve
BACKPROP	The Backpropagation Algorithm
CAV	Connected/Autonomous Vehicles
CBLOF	Cluster-Based Local Outlier Factor TML technique
CNN	Convolutional Neural Network
CONOPS	Concept of Operations
CPS	Cyber-Physical System
DATACENTER	The Yahoo Datacenter anomaly detection dataset used for experimentation.
DA-GMM	Deep Autoencoding Gaussian Mixture Model
DNN	Deep Neural Network
DSN	Distributed Sensor Network
DSS	Decision Support System
ED-1D-CNN	Encoding-Decoding One-Dimensional Recurrent Neural Network
ED-RNN	Encoding-Decoding Recurrent Neural Network
ES	Earth Science
FAD	Federated Anomaly Detector
FFN	Feed-Forward Neural Network
FL	Federated Learning
FN / FNR	False Negative / False Negative Rate
FP / FPR	False Positive / False Positive Rate
FRAUD	The credit card fraud dataset used for experimentation
GAN	Generative Adversarial Network
GAT	Global Air Traffic
GC	Global Contraband
GP	Global Pandemics
GPU	Graphical Processing Unit
HBOS	Histogram-Based Outlier Score TML technique
IC	Industrial Control
IDS	Intrusion Detection System
IF	Isolation Forrest TML technique
IoT	Internet-of-Things
K-L	Kullback-Leibler Divergence
k-NN	k Nearest Neighbor TML technique
LOF	Local Outlier Factor TML technique
LSTM	Long Short-Term Memory
MCD	Minimum Covariance Determine TML technique
NLP	Natural Language Processing
OC_SVM	One-Class Support Vector Machine TML technique
PCA	Principal Components Analysis TML technique
RE	Reconstruction Error
ROC	Receiver Operating Characteristic
RNN	Recurrent Neural Network
SAD	Stream Anomaly Detector
SCADA	Supervisory Control and Data Acquisition
SDA	Shallow-Deep Autoencoder
SGD	Stochastic Gradient Descent
SNS	Social Networking Stream
STADE	Spatiotemporal Anomaly Detection Environment
SVM	Support Vector Machine
SWAT	The Secure Water Treatment Testbed dataset used for experimentation
T-SNE	t-distributed Stochastic Neighbor Embedding
TML	Traditional Machine Learning
TN / TNR	Ture Negative / True Negative Rate
TP / TPR	True Positive / True Positive Rate
USGS	United States Geological Survey
VAE	Variational Autoencoder
WADI	The Water Distribution Testbed dataset used for experimentation.

CHAPTER 2 – ANOMALIES AND ANOMALY DETECTION QUICK LOOK

2.1 Anomaly Detection Terminology

There are three types of anomalies: (1) point, (2) contextual, and (3) collective. Point anomalies occur when a sample is unusual with respect to one or more other samples. Contextual anomalies occur when a sample is unusual when viewed within a specific, operational context. A significant change in a sensor value over two adjacent periods or two adjacent regions, for example, might indicate a contextual anomaly. Collective anomalies occur when one or more samples are unusual with respect to the entire dataset. Point anomalies are more extensively studied and more straightforward to identify than collective and contextual anomalies.

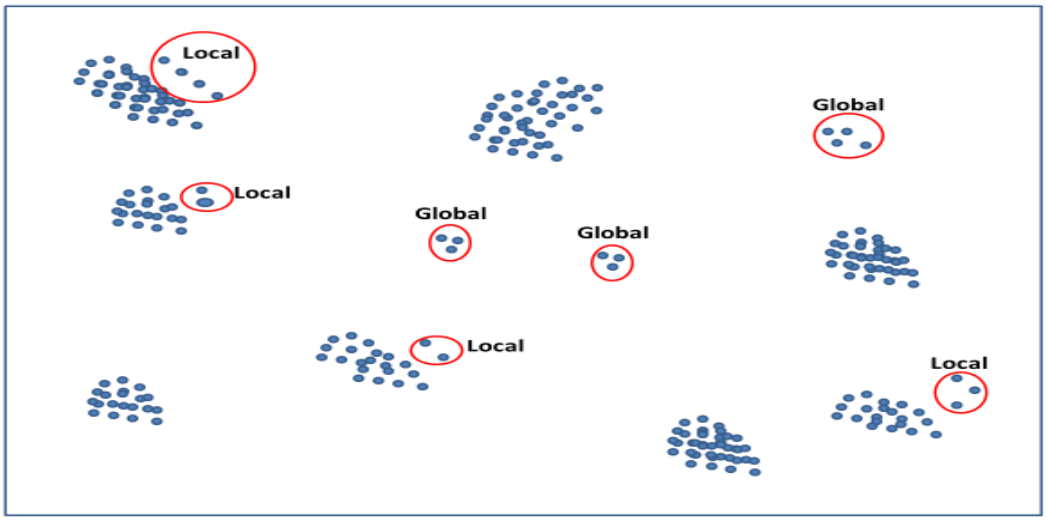


Figure 1: Local vs. Global Anomalies

Consider Figure 1, which illustrates the distinction between local and global anomalies. Local anomalies are identified with respect to the nearest neighbors, while global anomalies are identified with respect to the entire dataset. Therefore, local anomalies are typically point or contextual, while global anomalies are collective. However, there is not a one-to-one

relationship between the definitions of local and global anomalies, and point, contextual, and collective anomalies. Local anomalies often occur in ecology and other biological domains.

Note the subtle distinction in the literature between an outlier, anomaly, and a novelty. An outlier is a data point that has a low probability of occurrence but is not necessarily anomalous. For example, random noise may cause outliers in data. An anomaly has an even lower probability than an outlier so that the definition is one of degree; the threshold is determined by the characteristics of the domain and the use case. A novelty is an infrequent event. However, the distinction between an outlier, anomaly, and novelty is not critical to the discussion here.

There are many core causes of anomalies. Uncertainties can cause anomalies and outliers. Uncertainty can be a result of incomplete domain understanding, incomplete observability, stochasticity, incomplete modeling, or inappropriate model abstraction. Anomalies can also be an outcome of unidentified changes over time in normal behavior, or intentional changes in behavior such as malicious actions by adversaries.

2.2 Streaming Spatiotemporal Data

Streaming is defined as the digital process of receiving data through a network in a continuous flow. The time intervals between receipt may or may not be equal depending on the problem domain. Stream processing is the term used synonymously with the term ‘online’ and is the opposite of batch processing. With streaming, each sample is processed sequentially with the model or algorithmic parameter updates on-the-fly. Anomaly scoring is also continuous; anomaly designations are made immediately at the time of receipt, and there is no revision of past designations. Upon receipt, samples are stored in local volatile memory, on disk in

persistent storage, transmitted via a network and combined with other sites in a central database server, or discarded on the spot.

Streaming samples may be unordered, time-dependent, geographic-dependent, or spatiotemporal. With unordered streams, each sample is independent of the next; the sequential nature of the stream is unimportant. A time-dependent stream, which is also known under the moniker of time-series, includes a time dimension. A geographic-dependent stream includes a unique location or geographic dimension. Examples of location or geographic features include a latitude and longitude or a computer internet protocol (IP) address. In many statistical models, larger geographic regions (e.g., North America) may be incorporated by one-hot or dummy variable encoding. The combination of time dependence and location significance in a data stream is the working definition of spatiotemporal.

Note that in non-engineering disciplines, spatiotemporal is referred to as a 'time series of cross-sections' (e.g., economics) or sequences (e.g., bioinformatics). However, not all sequences are logically streamed. A linearly ordered string such as a DNA encoding is ordered but not conceptually transmitted sequentially through a network and hence does not fit into the definition of streaming.

There are numerous application examples of streaming spatiotemporal data in the anomaly detection literature. The domain that has received the most research attention is network intrusion detection systems (IDSs). An IDS monitors the inbound and outbound network packets in real-time for malicious activity (e.g., viruses or worms) and policy violations (e.g., network traffic originating from a prohibitive source). An IDS detects anomalies based on individual packet snapshots or adjacent packet changes. Other examples of streaming spatiotemporal applications include global air traffic, global earthquake monitoring, and

Twitter™ social networking tweets. These latter examples are the focus of the case studies presented in Chapters 9-11.

Consider Figure 2, which illustrates the example of a temporal sensor stream from two locations without a geographic time lag. A contextual and point anomaly would be detected around periods 15 and 29 in both Region #1 and Region #2 due to the sudden spike down in the temporal neighborhood. However, there would be no collective anomaly because the value pattern repeats not only within a region but also between regions. These types of patterns may be easy to recognize with two regions but more challenging to identify with multivariate, multi-regional data.

Figure 3 illustrates a related example of a sensor network with two regions. Is this example, the sensors produce values that are identical in both regions; however, there is a five-time unit offset in region two. Both regions have the same shape of sensor values, so from a collective perspective, when multiple regions are analyzed jointly, no anomalies exist if an interregional lagged response is expected.

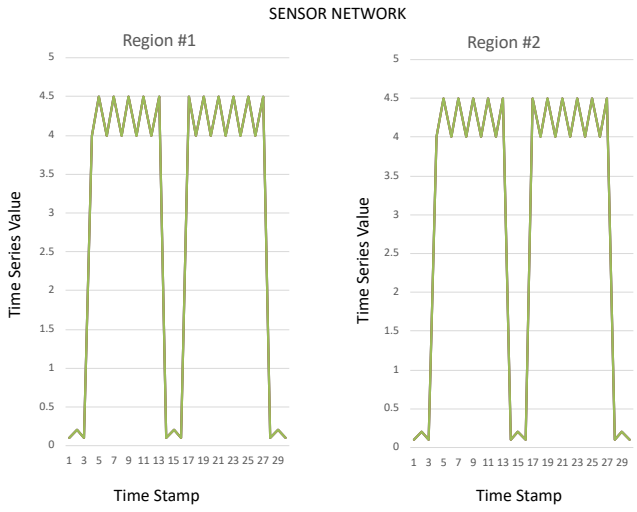


Figure 2: Sensor Network No Lag

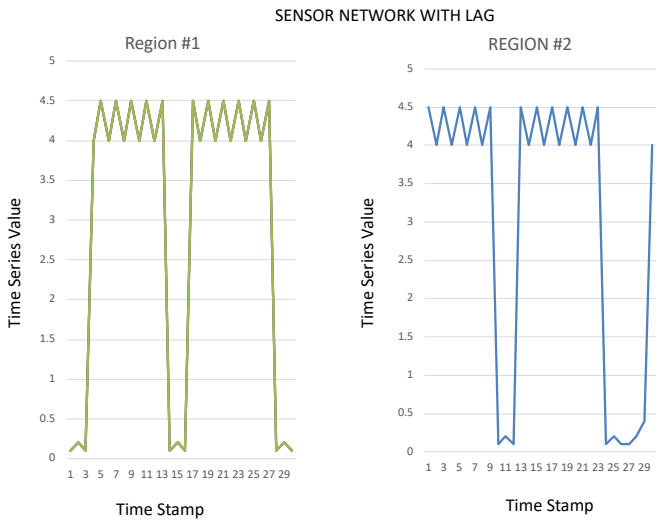


Figure 3: Sensor Network with Lag

Some anomaly detection techniques can be modified to support high-tempo streaming data even with slower executing algorithms. Techniques that can process sub-windows of the entire stream can be adapted for streaming sequence data by assigning an independent anomaly score to each sub-window. Scores are based on only the samples within the window. The overall anomaly may be designated when a sequence of scores is anomalous. The adaptation of existing anomaly detection techniques is discussed in Chapter 3 (TML models) and Chapter 5 (DNN models).

2.3 Supervised, Semi-Supervised, Self-Supervised, and Unsupervised Learning

Studies of anomaly detection employ supervised, semi-supervised, self-supervised, or unsupervised techniques. Supervised learning requires labeled training instances. Semi-supervised learning uses a combination of labeled and unlabeled data and is required in cases where there are insufficient labeled instances. Examples of supervised and semi-supervised TML techniques include support vector machines, decision trees, random forests, and multiple variants of regression analysis. DNNs formulated as a supervised learning problem are also

popular in the anomaly detection literature. For example, DNN-based fraud detection models use supervised learning because of the ready availability of financial transactions from financial institutions. Similarly, DNN-based network spam models use supervised learning because of the availability of attack data from commercial and internet providers. However, in most other domains, labeled datasets are not readily available.

Self-supervised approaches replicate the input data probabilistically through a model; the target is the input sample. Principally, self-supervised techniques are supervised since training occurs with a target variable, although labeled data are not used. An example of a self-supervised technique is the various flavors of autoencoders. Autoencoding and other techniques for self-supervision in the context of anomaly detection are discussed in detail in Chapter 6.

Unsupervised techniques are based on unlabeled data and assume that the training data includes both anomalous and non-anomalous. Some authors note the distinction between the definition of outlier detection and novelty detection. Outlier detection assumes that the training data may include anomalies, while novelty detection assumes that training data does not include anomalies. In any event, by definition, an anomaly is a rare event; the inclusion of infrequent anomalies in unsupervised training datasets usually does not introduce bias.

In summary, the focus of the research here is on self-supervised or unsupervised techniques (used interchangeably), where the assumption is that the training data may contain anomalies but at a low, insignificant frequency. Chapter 4 addresses unsupervised TML techniques, while Chapter 6 addresses unsupervised and self-supervised DNN techniques. Note that while unsupervised or self-supervised techniques do not require labeled anomaly data, labels are still required on the test data in order to measure the efficacy of the algorithm. Section 2.5 below discusses algorithm performance evaluation.

2.4 Anomaly Scoring and Labeling

At the core of anomaly detection is a classification problem. Labeling is the process of assigning a score or metric and specifying a decision function that maps the score or metric to a binary anomaly classification. While the classification is binary, the anomaly score will also indicate the probability of abnormality. Labels may be assigned to a single instance in the case of point anomalies, or a set of points, in the case of contextual or collective anomalies.

Each algorithm has a unique approach to anomaly scoring and the classification function. One approach is to rank-order the anomaly scores from low-to-high. Assuming that the highest scores are the most anomalous, and if there are x percent anomalies identified in the training data, then the top x percent of the anomaly scores in the test data are designated as anomalous. Other approaches are based on probability cut-off values, various distance measures, and asymmetric risk objectives. Scoring and labeling of streaming data occur continuously and in real-time.

Anomaly labeling may be multi-step. For example, an anomaly might be defined as a temporal sequence that produces a point anomaly in three consecutive periods in two or more regions. Moreover, if the risk profile is asymmetric, the underprediction of anomalies may introduce a higher risk than over prediction. An algorithm that consistently overpredicts this anomaly over four consecutive periods may be preferable to another algorithm that consistently underpredicts. These complexities make multi-step labeling challenging to generalize and highly domain-dependent.

There are four related approaches to multi-step anomaly labeling that are depicted in Figure 4. Approach (a) is to estimate the required number of steps using a one-step model. For example, for a three-step anomaly detection problem, three separate unsupervised models are

estimated concurrently. Approach (b) is a variant of the first approach. A separate model is estimated for each step, but the label from the previous step is used as input into the model for the next step. With this approach, the models are estimated sequentially. Approach (c) is to use a single model that is capable of estimating multiple steps. For example, a single DNN could be formulated as multiple nodes in the output layer. Approach (d) is through a recursive sequential approach. The recursive strategy involves using a one-step model where the label generated from the previous step is used as an input into the prediction of the label in the next step. This approach is preferred because of the lower computation requirements vis-à-vis running a separate model for each step. Sequence-to-sequence recurrent neural networks (RNN) discussed in Chapter 4 are based on this approach.

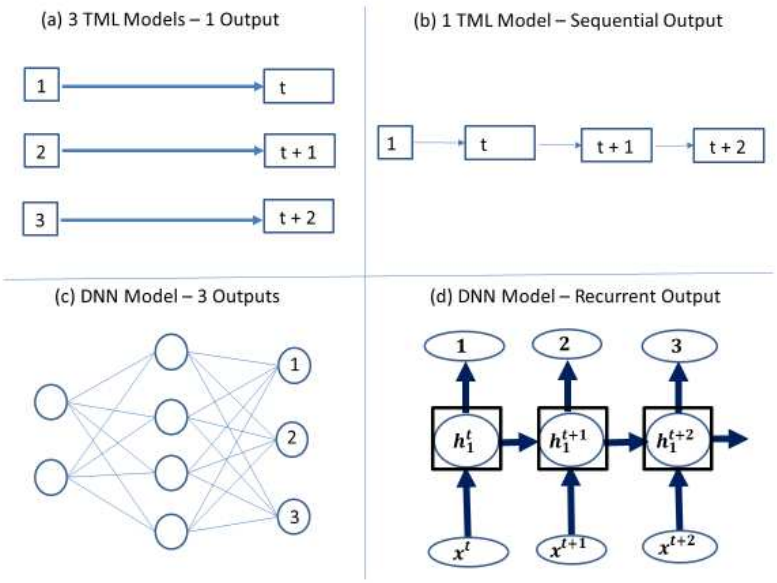


Figure 4: Multi-Step Labeling

2.5 Algorithm Performance Evaluation

The focus of this research is on unsupervised TML and DNN, techniques that do not require labeled data. However, without anomaly labels, how are algorithms evaluated? The

answer is that labeled data is still required for evaluation purposes; the model parameters are estimated using training data without labels, but the performance of the model is measured with evaluation data that includes anomaly labels. Model parameters are applied to the evaluation data, anomaly predictions made, and then compared against the actual labels to evaluate performance. These performance measures relevant to the evaluation data are discussed below.

A couple of critical assumptions should be noted here. First, in the anomaly detection literature, anomalies are deemed 'positive' outcomes, while non-anomalies are 'negative' outcomes. Second, the assumption is that anomalies are a rare event and are only a small percentage of the overall population. Note that predictive accuracy is an inappropriate measure of performance in anomaly detection studies because of the skewed population. If, for example, there were only one percent (1%) anomalies in the population, a model that predicted one-hundred percent (100%) non-anomalies would always be ninety-nine percent (99%) accurate.

2.5.1 Confusion Matrix

Table 3 displays an example of the confusion matrix used in classification studies. A confusion matrix is a two-by-two table that allows the comparison of the algorithmic performance on test data. In this table, each row indicates the actual anomaly label, while each column represents the predicted label. Therefore, the values down the diagonal represent perfect classifiers, 'true negatives' (TN), and 'true positives' (TP). Because of the data imbalance, most of the samples are expected to land in TN. The upper right quadrant includes false-positives (FP), which are non-anomalies misclassified as anomalies, and the bottom left quadrant includes 'false negatives' (FN), which are anomalies misclassified as non-anomalies. Therefore, the total number of misclassifications are FP+FN. With highly imbalanced data, classification accuracy is not an appropriate metric for success.

Note that with supervised learning, the output of a DNN is an anomaly prediction. With anomaly predictions, the calculation of the confusion matrix is straightforward since the anomalous population within the training dataset is known. The confusion matrix is based on a threshold probability such as .5. With unsupervised learning, the anomalous population within the training dataset is not known. Unsupervised techniques produce rank order anomaly scores and do not produce direct probability estimates. For this reason, the confusion matrix under supervised learning is only one example of a continuum of matrices based on the anomaly score threshold value.

Table 3: Confusion Matrix

Confusion Matrix		**Predicted**	
		Normal	**Anomalous**
Actuals	**Normal**	True Negative (TN)	False Positive (FP)
	Anomalous	False Negative (FN)	True Positive (TP)

2.5.2 Precision Recall Tradeoff

Let the decision function scoring threshold be denoted by *t*. Increases in the threshold *t* result in fewer anomalies being classified in the test data. There is no right or wrong value for the scoring threshold value *t*; this value depends on the objectives of the model or study.

The trade-off between failing to identify true anomalies (false negatives) and over-identifying anomalies (false positives) is measured in terms of precision and recall, a standard

approach to testing of classifiers. Precision is defined as the fraction of predicted anomalies (TP+FP) that are accurately predicted (TP):

$$\text{Precision}(t) = \frac{TP}{TP + FP} \tag{2.1}$$

Note that as the decision threshold *t* increases, the size of *FP* decreases as fewer anomalies are predicted. Moreover, the precision metric in equation (2.1) is not monotonic in *t* since both the numerator and denominator are a function of *t*.

Recall is defined as the fraction of actual anomalies (TP+FN) that were accurately predicted (TP). Recall is also known as the true positive rate (TPR) and is shown in equation (2.2):

$$Recall(t) = TPR = \frac{TP}{TP + FN} \tag{2.2}$$

The F_1 metric, which is also known as the F-measure or F-score, combines precision and recall into a single metric, the harmonic average, as shown in equation (2.3). This metric is useful when comparing two classifiers, is a measure of the overall accuracy of the algorithm, and is the harmonic average of the precision and recall. An F_1 close to one indicates both high precision and high recall:

$$F_1 = 2 * \frac{Precison\ x\ Recall}{Precision + Recall} \tag{2.3}$$

The Precision-Recall Tradeoff curves shown in Figure 5 illustrates the relationship between precision and recall metrics at various levels of the trade-off *t*. The two curves are different viewpoints of the same data. The left-side curve incorporates the threshold *t* implicitly while the right-side cure incorporates the threshold *t* as the x-axis. As the threshold *t* increases, fewer anomalies are classified, so that the false positives decrease, increasing precision. However, precision may also go down at some threshold *t* per equation (2.1), so the curve if

often concave. Recall, however, is always a monotonic decreasing function of the threshold *t*. An increase in threshold *t* will result in fewer predicted anomalies increasing false negatives. Given equation (2.2), an increase in false negatives will always result in a decrease in the recall metric.

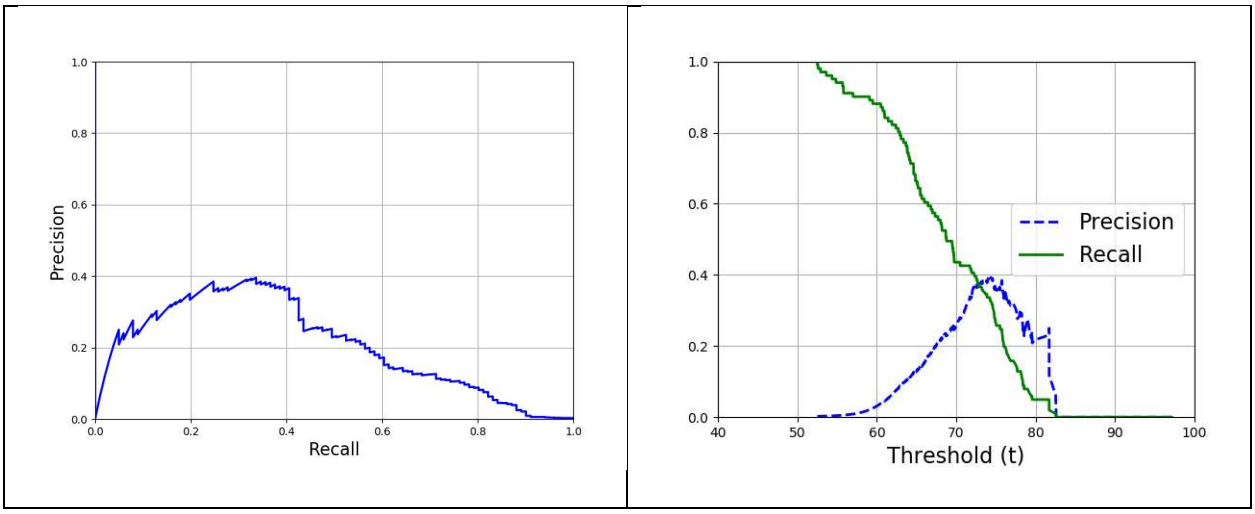


Figure 5: Precision-Recall Tradeoffs Example

2.5.3 The Receiver Operating Characteristic (ROC) Curve

The final evaluation technique is the Receiver Operating Characteristic (ROC) Curve; an example is shown in Figure 6. Similar to the precision-recall curves, each point on the ROC is calculated by increasing the threshold *t* by a small amount. Using a ROC curve, the false positive rate FPR(*t*), given by equation (2.4) is graphed on the x-axis, and the true positive rate TPR(*t*), which is the same as recall and was given by equation (2.2), is graphed on the y-axis.

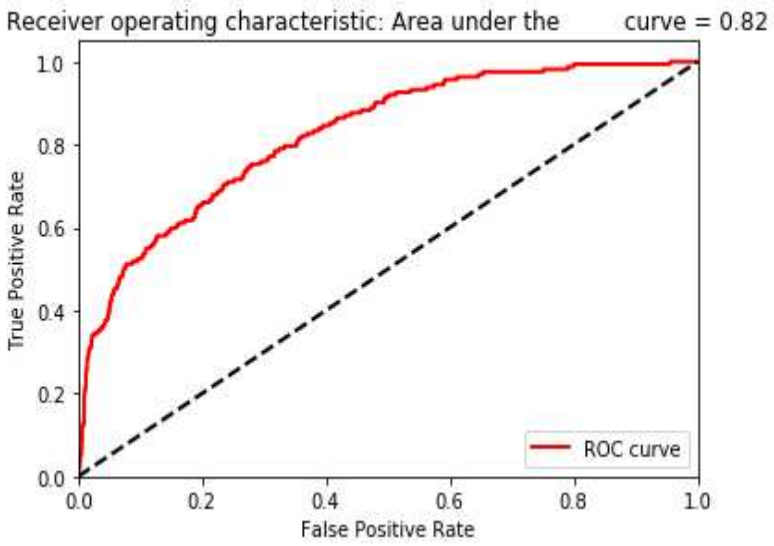


Figure 6: Receiver Operating Characteristic (ROC) Example

$$(FPR) = \ False\ Positive\ Rate(t) = \frac{FP}{TN + TP} \tag{2.4}$$

One statistic useful to compare the performance of different algorithms is the area under the Receiver Operating Characteristic (ROC) curve or AUC. If one anomaly and one non-anomaly sample are randomly selected from the dataset, AUC represents the likelihood that the algorithm will assign a higher predicted probability on the anomaly. A random classifier will have a ROC AUC equal to .5, while a perfect classifier will have a ROC equal to 1. The higher the AUC, the better is the performance of the algorithm. Therefore, the ROC AUC is the probability that a binary classifier will predict that a random positive (anomalous) sample is a higher probability than a random negative (non-anomalous) sample. Top-performing anomaly detection algorithms should produce AUCs above approximately .75 at a minimum.

CHAPTER 3: EXPERIMENTATION DATASETS

3.1 Introduction

Experimentation was conducted using four highly specialized anomaly detection datasets from different application domains. All experimentation was performed in unsupervised mode. Unsupervised mode means that labels were not used to train the model parameters and only used to evaluate the algorithm against the test data. While the TML and DNN algorithms to be discussed supports unsupervised or semi-supervised learning, there is still a practical need for labeled test data to compare, evaluate, and score alternative techniques. The anomaly detection performance metrics used to score were discussed in the previous chapter, Section 2.5. Evaluation metrics must be carefully selected because the substantial imbalance between the number of non-anomalous samples and the number of anomalous samples will skew the results.

Chapter 2 also discussed the nature of contextual and collective anomalies that can be identified only after the analysis of the complete dataset. Most datasets, however, only identify point anomalies in isolation and that occur at an instance in time. Algorithms designed to detect point anomalies may still be used for the study of contextual or collective anomalies by including features that capture or proxy time and location features.

Temporal sequence streaming data include a time-stamp feature associated with each sample. Ideally, the processing architecture and algorithm under experimentation should be exercised under the operational conditions of streaming data where samples are processed in real-time. In practice, this approach is rarely possible because datasets are collected from testbeds over many days or weeks of operation, and replicating this timing in experimentation is not feasible. Sensor readings from the water treatment and water distribution testbeds (described below) were captured over an extended period. If the processing architecture and algorithm can

execute ‘faster than real-time,’ then speedup will not introduce artificialities into the model training process. However, if the processing algorithm executes slower than real-time, artificialities are introduced that may compromise the validity of the experimentation and resultant conclusions.

Note that anomalies (e.g., cyberattacks) can extend over long periods. Anomaly detection algorithms may identify those attacks for only part of the attack window or for a period that extends beyond the completion of the attack. Whether the algorithm receives credit for correctly identifying those attacks as anomalies depends on the data collection processes and the requirements of the case study. Early recognition of a cyberattack may be more beneficial than belated recognition because corrective measures can be implemented proactively to mitigate damage.

3.2 Feature Scaling

Feature scaling is the process of transforming the range of values into a standard or comparable scale. The primary reason for feature scaling is that TML and DNN (i.e., stochastic gradient descent (SGD) algorithms) are more stable and converge faster with features that have a standard scale. Particularly with respect to SGD, large input values might cause numerical overflows and other non-transparent estimation problems. Target features (i.e., the dependent variable) and binary or one-hot encoded features, however, are not scaled.

There are two conventional approaches to scaling: min-max scaling and standardization. With min-max scaling, features are transformed to the range from zero to one. With standardization, the sample mean value is subtracted from each value and divided by the sample standard deviation, resulting in a zero mean and unit variance. Standardization does not bound

vales to a specific range like min-max scaling but is less influenced by extreme outliers in the data. Min-max scaling and standardization are adopted throughout the remaining chapters.

3.3 Datasets for Experimentation

Four multivariate datasets were identified for experimentation: (a) the Credit Card Fraud (FRAUD) dataset, (b) the Secure Water Treatment (SWAT) dataset, (c) the Water Distribution (WADI) dataset, and (d) the Yahoo DATACENTER dataset. The characteristics of these datasets are briefly discussed below. Note that each dataset is time-ordered but does not include a spatial component. Unfortunately, publicly available spatiotemporal anomaly detection experimentation datasets are not readily available. The purpose of experimentation here is to explore the variety of algorithms that could be adapted for spatiotemporal anomaly detection. The concept of spatiotemporal anomaly detection is further addressed with the STADE specification provided in Chapter 8 and the three STADE case studies provided in Chapters 9-11.

3.3.1 Credit Card Fraud (FRAUD)

The FRAUD dataset consists of credit card transactions over two days in 2013 in Europe. A fraudulent transaction is an anomaly. In total, 284,807 transactions were processed with 492 transactions deemed as fraudulent (and 284315 non-fraudulent), resulting in an anomaly rate of 0.17 percent. Included are twenty-eight numerical explanatory but undefined features; however, each feature has been obfuscated by Principal Components Analysis (PCA) for confidentiality so that the features are not interpretable. The two features that were not transformed using PCA are the time and the transaction amount. For an overview of fraud detection techniques, see Kou, Lu, and Sirwongwattana [36].

The FRAUD dataset is not a real temporal dataset. While the time feature contains the seconds elapsed from the start of data collection, the value has no impact on the algorithmic

performance since the entities responsible for the transaction cannot be uniquely identified. Nevertheless, the dataset was selected because of the quality of the data and the extensive use in the anomaly detection literature.

Table 4 displays the FRAUD raw statistics by anomaly designation. All features except time and transaction amount (in EURO currency) have been transformed by PCA so that the values are centered around zero. Note that transaction amounts are, on average, higher for fraudulent samples than with routine transactions. However, the non-anomalous transactions include an apparent outlier (maximum of 25691€), which is significantly above the maximum anomalous transaction (2125€). For many features, the standard deviation of the anomalous transactions is measurably higher than the standard deviation of non-anomalous features.

Table 4: FRAUD Statistics

Feature	Non-Anomalous				Anomalous (Fraudulent)			
	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max
Time	94838	47484	0.00	172792	80746	47835	406	170348
V1	0.00	1.92	-56.40	2.45	-4.77	6.78	-30.5	2.13
V2	0.00	1.63	-72.71	18.90	3.62	4.29	-8.4	22.0
V3	0.01	1.45	-48.32	9.38	-7.03	7.11	-31.10	2.25
V4	0.00	1.39	-5.68	16.87	4.54	2.87	-1.31	12.11
V5	0.00	1.35	-113.74	34.80	-3.15	5.37	-22.10	11.09
V6	0.00	1.32	-26.16	73.30	-1.39	1.85	-6.40	6.47
V7	0.00	1.17	-31.76	120.58	-5.56	7.20	-43.55	5.80
V8	0.00	1.16	-73.21	18.70	0.57	6.79	-41.04	20.00
V9	0.00	1.08	-6.29	15.59	-2.58	2.50	-13.43	3.35
V10	0.00	1.04	-14.74	23.74	-5.67	4.89	-24.58	4.03
V11	0.00	1.00	-4.79	10.00	3.80	2.67	-1.70	12.01
V12	0.01	0.94	-15.14	7.84	-6.25	4.65	-18.68	1.37
V13	0.00	0.99	-5.79	7.12	-0.10	1.10	-3.12	2.81
V14	0.01	0.89	18.39	10.52	-6.97	4.27	-19.21	3.44
V15	0.00	0.91	-4.39	8.87	-0.09	1.04	-4.49	2.47
V16	0.00	0.84	-10.11	17.31	-4.13	3.86	-14.12	3.13
V17	0.01	0.74	-17.09	9.25	-6.66	6.97	-1.34	6.79
V18	0.00	0.82	-5.36	5.04	-2.24	2.89	-9.49	3.79
V19	0.00	0.81	-7.21	5.59	0.68	1.53	-3.68	5.22
V20	0.00	0.76	-54.49	39.42	0.37	1.34	-4.12	11.05
V21	0.00	0.71	-34.83	22.61	0.71	3.86	-22.79	27.20
V22	0.00	0.72	-10.93	10.50	0.01	1.49	-8.88	8.36
V23	0.00	0.62	-44.80	22.52	-0.04	1.57	-19.25	5.46
V24	0.00	0.60	-2.83	4.58	-0.10	0.51	-2.02	1.09
V25	0.00	0.52	-10.29	7.51	0.04	0.79	-4.78	2.20
V26	0.00	0.48	-2.60	3.51	0.05	0.47	-1.152	2.74
V27	0.00	0.39	-22.56	31.61	0.17	1.37	-7.26	3.05
V28	0.00	0.32	-15.43	33.84	0.07	0.54	-1.86	1.77
Amount	88.29	250.1	0.00	25691.16	122.21	256.68	0.00	2125.87

Figure 7 displays transaction amounts by time. Note that there are a few transaction outliers, but as previously noted, the FRAUD dataset is useful only for point anomaly detection.

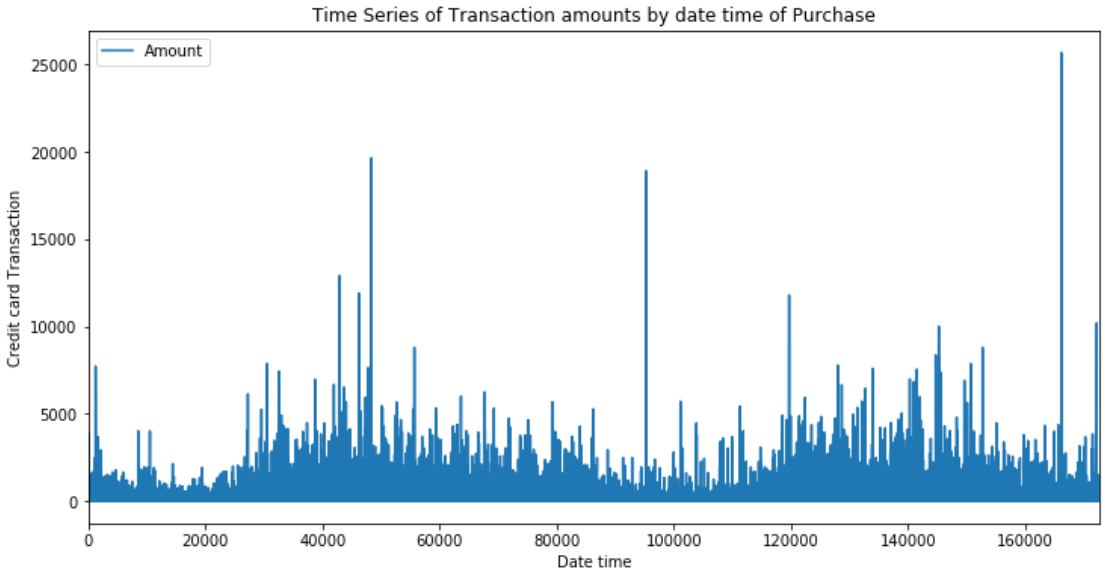


Figure 7: FRAUD - Time Series of Transactions by Amount

3.3.2 Secure Water Treatment (SWAT) Testbed

SWAT is an operational testbed for water treatment research and development in support of the Singapore Public Utility Board. SWAT is operated by the iTrust Centre for Research in Cyber Security. The testbed was designed to simulate and test the effects of cyberattacks on a water treatment facility. The attack points include various sensors (e.g., water level) and actuators (e.g., water pumps). Figure 8 provides a picture of the SWAT testbed that produced the dataset.

Experimentation data was collected over eleven (11) days of continuous operation. The first seven (7) days captured standard operation data, while the final four (4) days captured data under various simulated cyberattack scenarios. Therefore, the first seven (7) days of data are used for model training, while the last four (4) days of data are used for model testing. In total,

there were 495,000 records processed during normal operations and 449,919 records processed during the periods of attack, of which 54,584 records were labeled as anomalous. Values from all fifty-one (51) sensors, network traffic, and actuators were recorded. Thirty-six (36) distinct attacks of various types were initiated and recorded, ranging in length from two minutes to twenty-five minutes. Examples of attack impacts include water tank overflow, chemical discharge, and output valve shutdown.

Figure 8: Secure Water Treatment (SWAT) Testbed

Table 5 displays the SWAT statistics by anomaly designation. Note that with some features, anomalous samples exhibit lower mean values than non-anomalous samples (e.g., FIT101), while with other features, the reverse is true (e.g., LIT101). The standard deviations are also measurably different between non-anomalous and anomalous samples for some features (e.g., AIT203). Since the anomaly detection algorithms are multivariate that capture complex interrelationships, conclusions cannot be drawn simply by inspecting individual feature values or variances. Perhaps more critical, the sequence of values rather than the point values may be the most critical determinant of an anomaly. As previously noted, techniques for point anomaly detection are not typically useful for the detection of contextual or collective anomalies that occur over some time.

Table 5: SWAT Statistics

Feature	Non-Anomalous				Anomalous			
	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max
Time	457036	272573	0	944918	723844	95517	496754	940190
FIT101	1.84	1.13	0	2.76	0.78	1.17	0.00	2.70
LIT101	588.79	118.45	120.62	888.17	727.41	135.35	189.82	925.03
MV101	1.71	0.46	0.00	2.00	1.30	0.46	0.00	2.00
P101	1.75	0.43	1.00	2.00	1.27	0.44	1.00	2.00
P102	1.00	0.01	1.00	2.00	1.05	0.22	1.00	2.00
AIT201	240.53	35.58	168.03	272.52	202.22	26.58	168.80	265.18
AIT202	8.44	0.11	6.00	8.988	8.54	0.16	6.00	8.70
AIT203	334.87	40.99	285.33	567.46	337.13	21.16	287.95	370.54
FIT201	1.83	1.05	0.00	2.82	0.68	1.12	0.00	2.82
MV201	1.74	0.44	0.00	2.00	1.28	0.45	0.00	2.00
P201	1.06	0.23	1.00	2.00	1.01	0.11	1.00	2.00
P203	1.74	0.43	1.00	2.00	1.26	0.44	1.00	2.00
P204	1.00	0.00	1.00	1.00	1.00	0.03	1.00	2.00
P205	1.70	0.45	1.00	2.00	1.27	0.44	1.00	2.00
P206	1.00	0.00	1.00	1.00	1.00	0.03	1.00	2.00
DPIT301	16.71	6.72	0.00	45.00	8.41	10.24	0.01	45.00
FIT301	1.84	0.80	0.00	2.37	0.66	1.00	0.00	2.35
LIT301	901.04	85.05	123.81	1201.00	964.95	109.63	364.38	1201.00
MV301	1.00	0.11	0.00	2.00	1.01	0.13	0.00	2.00
MV302	1.80	0.41	0.00	2.00	1.29	0.46	0.00	2.00
MV303	1.02	0.17	0.00	2.00	1.01	0.15	0.00	2.00
MV304	1.02	0.19	0.00	2.00	1.63	0.49	0.00	2.00
P301	1.00	0.04	1.00	2.00	1.00	0.00	1.00	1.00
P302	1.83	0.37	1.00	2.00	1.29	0.45	1.00	2.00
AIT401	135.41	42.58	0.00	148.85	148.80	0.00	148.76	148.85
AIT402	161.71	14.89	141.11	327.83	250.77	85.33	140.83	333.811
FIT401	1.71	0.07	0.00	1.74	0.66	0.82	0.00	1.74
LIT401	881.54	89.67	130.38	1003.93	497.87	309.65	243.01	1002.58
P402	1.99	0.04	1.00	2.00	1.41	0.49	1.00	2.00
P403	1.00	0.00	1.00	2.00	1.00	0.00	1.00	1.00
UV401	1.99	0.04	1.00	2.00	1.39	0.48	1.00	2.00
AIT501	7.84	0.05	7.41	8.30	7.65	0.16	7.43	8.25
AIT502	151.55	13.52	129.83	272.85	188.29	41.89	131.81	271.03
AIT503	265.84	6.06	244.90	297.96	266.58	4.17	244.87	281.53
AIT504	13.02	5.63	7.34	442.46	16.62	19.78	11.18	255.00
FIT501	1.72	0.07	0.00	1.75	0.70	0.83	0.00	1.75
FIT502	1.27	0.05	0.00	1.36	0.52	0.62	0.00	1.35
FIT503	0.73	0.31	0.00	0.76	0.29	0.35	0.00	0.74
FIT504	0.30	0.01	0.00	0.31	0.11	0.14	0.00	0.31
P501	1.99	0.04	1.00	2.00	1.39	0.48	1.00	2.00
PIT501	250.77	10.53	8.89	264.64	106.71	115.68	9.46	253.12
PIT502	1.14	0.25	0.00	3.66	0.46	0.63	0.00	1.89
PIT503	189.63	8.30	3.10	200.63	78.50	89.25	3.14	191.34
FIT601	0.01	0.15	0.00	1.80	0.01	0.13	0.00	1.74
P602	1.00	0.09	1.00	2.00	1.00	0.07	1.00	2.00

Figure 5 shows the time times of over normal operations of sensor LIT01, while figure 6 shows this same sensor values throughout the cyberattack. Note the spikes in values downward indicate likely anomalies. However, given that cyberattacks occur over time, more complex lagged relationships between the attack trajectory and anomalous sensor readings are likely.

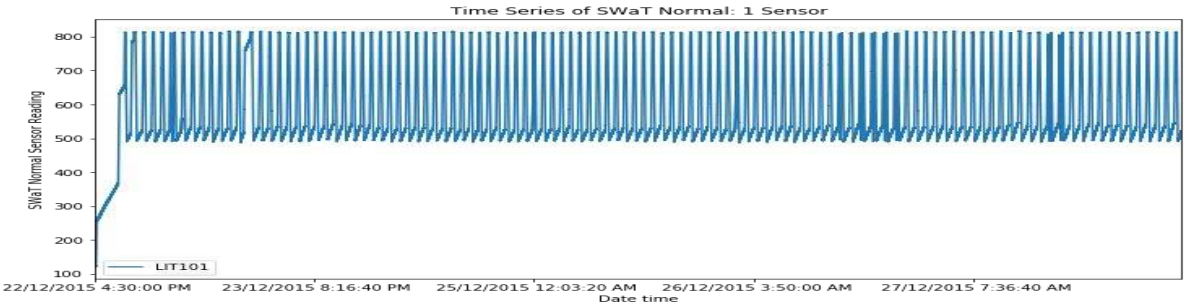


Figure 9: Normal Operations of Sensor LIT01

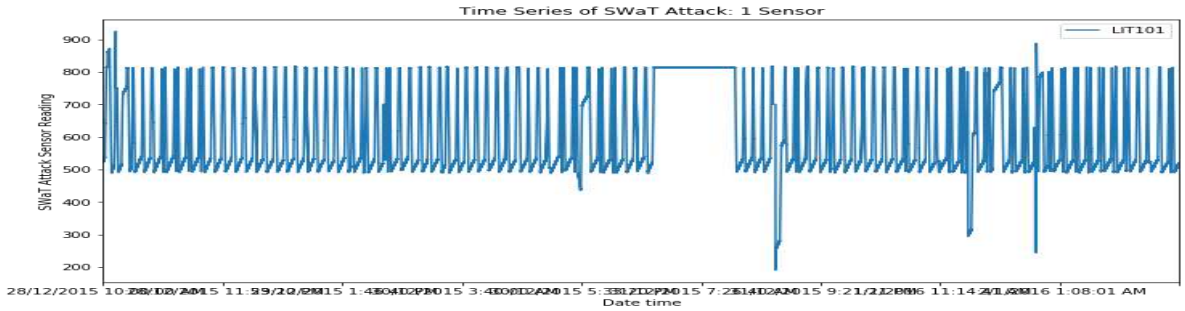


Figure 10: Under Attack of Sensor LIT01

3.3.3 Water Distribution (WADI) Testbed

The WADI testbed was designed by the iTrust Centre for Research in Cyber Security to simulate an unsecured water distribution network. In addition to the sensors and pumps similar to the SWAT testbed, the WADI testbed can simulate the effects of the cyberattacks resulting in water leakages and malicious chemical injections into the water supply. Figure 11 displays a picture of the WADI testbed.

Figure 11: Water Distribution (WADI) Testbed

The WADI dataset consists of sixteen (16) days of operation, with the first fourteen (14) days of normal operations and the last two (2) days of attack scenarios. A total of fifteen attacks were launched, ranging from 1.5 minutes to 30 minutes in length. The dataset recorded readings from the 103 sensors, actuators, and network devices in the testbed. However, some readings were constant throughout experimentation and were excluded from the analysis. For purposes of exposition, the devices are designated 'V1' through 'V127'.

Table 6 provides the complete descriptive statistics from the testbed of the features used in the analysis. Some of the features are continuous, while others are integers. For example, the values of the 'V1' and 'V2' sensors are significantly higher under attack than under normal operations. Figures 12 and 13 show the 'V1' sensor readings under normal operations and attacks. Note that both sensors appear to shut down periodically under typical and attack scenarios. At only one time point, does the 'V1' sensor appear to be anomalous with a significant spike upward. Overall, while some sensors, actuators, and devices appear to be affected by cyberattack activity, others do not. As with the SWAT testbed, complex interrelationships exist between the readings.

Table 6: WADI Statistics

Feature	Non-Anomalous				Anomalous			
	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Mas
V1	168.57	12.68	0.0	214.31	188.22	74.56	0.0	634.49
V2	0.62	0.06	0.0	6.0	0.98	1.38	0.0	6.0
V3	11.77	0.19	0.0	12.18	11.96	0.3	0.0	12.06
V4	483.8	25.3	0.0	526.53	447.11	22.93	0.0	480.53
V5	0.3	0.05	0.2	0.42	0.26	0.04	0.22	0.34
V6	0.52	0.85	0.0	2.08	1.48	0.81	0.0	2.5
V9	56.8	11.26	0.03	100.22	49.21	8.91	37.11	74.53
V10	1.27	0.45	0	2	1.68	0.51	0	2
V11	1.0	0.01	0	2	1.13	0.35	0	2
V12	1.0	0.01	0	2	1.13	0.35	0	2
V13	1.28	0.45	0	2	1.11	0.32	0	2
V14	1.27	0.44	1	2	1.72	0.45	1	2
V15	1.0	0.0	1	1	1.0	0.0	1	1
V16	1.27	0.44	1	2	1.72	0.45	1	2
V17	1.0	0.0	1	1	1.0	0.0	1	1
V18	1.23	0.42	1	2	1.41	0.49	1	2
V19	1.0	0.01	1	2	1.0	0.0	1	1
V20	2517.66	135.34	0.0	2777.41	2388.52	185.09	1033.12	2658.01
V21	63.98	36.94	7.53	100.0	47.79	35.28	8.79	100.0
V22	0.12	0.14	0.02	3.15	0.21	0.15	0.02	1.44
V23	0.22	0.12	0.09	0.51	0.28	0.13	0.1	0.48
V24	64.16	37.3	8.6	100.0	55.73	38.26	9.97	100.0
V25	0.11	0.12	0.02	2.94	0.16	0.14	0.04	1.51
V26	0.21	0.12	0.08	0.52	0.24	0.09	0.12	0.42
V27	67.42	38.4	4.11	100.0	69.94	39.32	6.73	100.0
V28	0.1	0.12	0.02	3.18	0.1	0.11	0.02	0.91
V29	0.21	0.12	0.06	0.52	0.25	0.09	0.11	0.38
V30	58.99	37.71	2.52	100.0	50.07	37.63	7.69	100.0
V31	0.12	0.14	0.02	3.04	0.23	0.17	0.02	1.97
V32	0.21	0.11	0.08	0.51	0.33	0.14	0.12	0.49
V33	61.8	39.39	4.5	100.0	60.34	40.78	5.64	100.0
V34	0.11	0.13	0.02	3.3	0.18	0.2	0.02	0.9
V35	0.22	0.14	0.09	1.15	0.39	0.13	0.12	0.51
V36	64.9	38.76	4.2	100.0	66.49	38.4	5.78	100.0
V37	0.1	0.12	0.02	3.16	0.09	0.12	0.02	1.55
V38	0.22	0.13	0.08	0.52	0.25	0.09	0.12	0.45
V39	0.51	0.94	0.0	3.67	0.92	1.09	0.0	2.29
V40	0.29	0.27	0.0	1.6	0.33	0.41	0.0	1.68
V41	0.21	0.52	0.0	5.15	0.53	0.7	0.0	3.24
V42	0.12	0.14	0.02	3.07	0.21	0.15	0.02	1.59
V43	0.11	0.12	0.02	2.95	0.16	0.14	0.04	1.5
V44	0.1	0.12	0.02	3.06	0.1	0.11	0.02	0.93
V45	0.12	0.14	0.02	3.01	0.23	0.17	0.02	1.76
V46	0.11	0.13	0.02	3.1	0.18	0.2	0.02	0.82
V47	0.1	0.12	0.02	3.12	0.09	0.12	0.02	1.55
V51	0.0	0.06	0	1	0.01	0.08	0	1
V52	0.0	0.04	0	1	0.0	0.0	0	0
V53	0.0	0.06	0	1	0.0	0.05	0	1
V54	0.0	0.05	0	1	0.0	0.0	0	0
V55	0.0	0.05	0	1	0.0	0.0	0	0
V56	0.0	0.04	0	1	0.0	0.0	0	0
V57	0.0	0.06	0	1	0.01	0.08	0	1
V58	0.0	0.05	0	1	0.0	0.04	0	1
V59	0.0	0.06	0	1	0.0	0.0	0	0
V60	0.01	0.09	0	1	0.0	0.0	0	0
V61	0.0	0.06	0	1	0.0	0.06	0	1
V62	69.65	0.58	68.51	71.55	69.9	0.28	69.09	70.34
V63	75.14	4.1	19.25	94.57	70.16	11.69	18.55	84.0
V64	0.0	0.23	0	100	10.96	28.33	0	100

---Continued on Next Page---

Feature	Non-Anomalous				Anomalous			
	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max
V65	11.1	15.55	0.0	100.0	18.22	27.17	0.0	100.0
V66	12.51	17.76	0.0	100.0	23.09	32.68	0.0	100.0
V67	16.8	20.91	0.0	100.0	35.65	34.79	0.0	100.0
V68	10.44	15.68	0.0	100.0	20.11	32.4	0.0	100.0
V69	13.61	17.44	0.0	100.0	21.4	26.09	0.0	100.0
V70	17.21	22.85	0.0	100.0	31.73	30.26	0.0	100.0
V73	1.22	0.44	0	2	1.36	0.56	0	2
V76	1.15	0.39	0	2	1.4	0.5	0	2
V78	1.47	0.5	0	2	1.16	0.37	0	2
V79	1.51	0.5	0	2	1.34	0.48	0	2
V80	1.57	0.5	0	2	1.5	0.5	1	2
V81	1.43	0.5	0	2	1.15	0.36	0	2
V82	1.5	0.5	0	2	1.38	0.49	0	2
V83	1.54	0.5	0	2	1.48	0.5	0	2
V86	3.48	8.11	-0.1	41.79	8.16	10.45	-0.02	39.45
V87	1.16	0.37	1	2	1.38	0.49	1	2
V88	0.03	0.01	-0.06	0.04	0.02	0.02	-0.03	0.03
V90	45.7	9.48	0.0	50.0	39.92	12.11	8.25	50.0
V91	0.22	0.42	0.01	3.82	0.4	0.49	0.01	1.9
V92	1.0	0.0	1.0	1.0	0.99	0.09	0.25	1.0
V93	152.33	2.05	123.88	163.33	147.58	5.83	129.75	158.26
V94	89.36	7.32	0.0	99.83	81.69	10.76	1.32	97.51
V95	0.22	0.42	0.01	3.82	0.4	0.49	0.01	1.9
V102	0.17	0.01	0.14	0.32	0.18	0.0	0.17	0.18
V103	0.0	0.05	0.0	7.9	0.0	0.06	0.0	1.74
V104	8.62	0.1	8.19	8.87	8.63	0.04	8.56	8.74
V105	482.3	7.68	457.31	502.0	472.84	3.47	464.91	479.34
V106	0.19	0.01	0.16	0.23	0.19	0.01	0.18	0.2
V107	543.04	2010.45	8.52	8128.0	3123.66	3941.87	8.72	8127.68
V108	8.59	0.12	6.73	8.87	8.64	0.05	8.57	8.74
V109	486.28	8.53	0.0	571.28	475.97	3.54	466.08	483.98
V110	0.0	0.02	0.0	0.16	0.0	0.0	0.0	0.0
V111	7301.73	2585.06	0.0	8295.95	8279.1	0.0	8279.1	8279.1
V112	8.99	8.75	0.0	53.74	9.77	8.48	0.0	34.97
V113	618.06	604.14	-2590.79	2831.28	563.14	585.26	-978.15	1766.92
V114	0.67	0.16	0.07	1.21	0.65	0.13	0.43	0.89
V115	0.52	0.26	0.0	1.16	0.41	0.31	0.0	0.99
V117	65.16	1.18	63.96	69.18	64.84	1.04	64.19	68.48
V125	62.96	5.88	45.4	147.29	65.8	6.94	58.97	138.2
V127	0.55	0.44	0.0	2.33	1.1	0.7	0.0	2.33

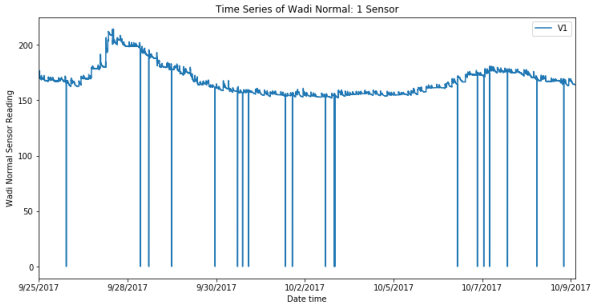


Figure 12: 'V1' Sensor - Normal Operations

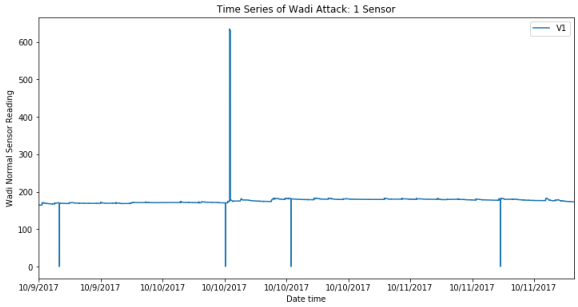


Figure 13: 'V1' Sensor - Under Attack

3.3.4 DATACENTER

The DATACENTER dataset consists of real and simulated data center network traffic made available for research purposes. While the anomalies in the simulated dataset were algorithmically generated, the anomalies in the real-traffic dataset were manually labeled and prone to human interpretation. For this reason, only the synthetically generated datasets are used. The dataset consists of one-hundred (100) different locations with approximately 1700 time-stamped samples per datacenter with seven (7) features. Because of the location and time components, this dataset is particularly desirable for spatiotemporal and streaming analysis since each dataset represents a physical datacenter.

Table 7 provides example statistics for the seven (7) features included in the dataset. This dataset includes pure time-series style features (e.g., seasonality) and has the smallest sample size vis-à-vis the other datasets. The core feature used to determine anomalies is 'value.'

Table 7: DATACENTER Statistics

Feature	Normal				Anomalous			
	Mean	Std Dev	Min	Max	Mean	Std Dev	Min	Max
Value	261.77	1295.46	-5332.44	6323.90	96.88	1852.58	-6171.31	6093.26
Changepoint	0.00	0.03	0.00	1.00	0.00	0.03	0.00	1.00
Trend	261.39	1202.06	-5040.00	5040.00	180.94	1209.76	-4695.00	4302.00
Noise	0.27	92.80	-802.36	902.05	0.75	95.79	-465.11	611.65
Seasonality1	-0.02	355.94	-998.00	998.00	-4.36	348.69	-998.00	848.00
Seasonality2	-0.03	264.91	-932.00	932.00	-2.05	274.31	-900.24	932.00
Seasonality3	0.17	179.71	-739.00	739.00	0.18	174.92	-642.51	665.81

Figure 14 displays the time-series of the 'value' feature for normal operations for datacenter #1, while Figure 15 provides the same 'value' feature but under attack operations. High and low spikes appear at different attack times points and are indicated in the dataset by the changepoint features. This dataset is likely better addressed by traditional univariate time-series algorithms but is included in determining if TML and DNN algorithms are effective detectors.

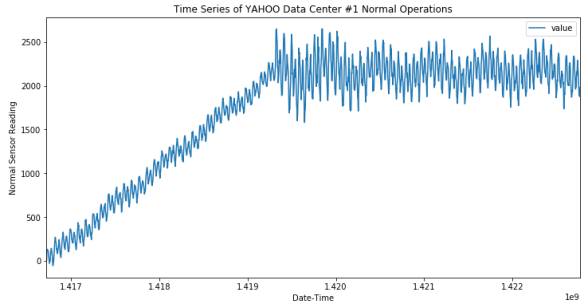


Figure 14: Datacenter #1 'Value' – Normal Ops

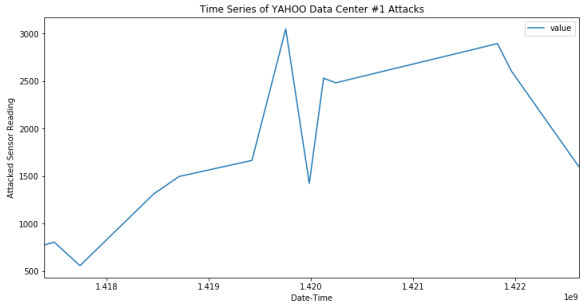


Figure 15: Datacenter #1 'Value' – Attack Ops

3.4 Experimentation Dataset Summary

Table 8 summaries the four data sources selected for TML and DNN experimentation:

Table 8: Comparison of Anomaly Detection Datasets

Attribute	**FRAUD**	**SWAT**	**WADI**	**DATACENTER****
Domain	Finance	Water Treatment	Water Distribution	Data Centers
Labeled Anomalies	492	54621	172801	4837
Data	Numeric	Numeric	Numeric	Numeric
Features	30	51	103	7
Temporal	Yes	Yes	Yes	Yes
Geospatial	No	No – Single Testbed	No – Single Testbed	Yes
Cyber Attacks	N/A	36	15	No
Attack Durations	N/A	2-25 minutes	1.5-30 minutes	N/A
Training Size	~227846	496800	1048571	~134400
Test Size	~56961	449919	172801	33600

N/A = Not Applicable **Across all 100 datacenters

The FRAUD dataset was the subject of one of many Kaggle online competitions to produce the best classification results in a particular problem domain using machine learning techniques. The SWAT and WADI datasets are relatively new to the anomaly detection DNN research community and have recently appeared in [37] and [38]. There are no known research studies that have analyzed the DATACENTER dataset.

CHAPTER 4: TRADITIONAL MACHINE LEARNING (TML) ALGORITHMS

4.1 Introduction

There is no universal consensus by anomaly detection researchers regarding the appropriate taxonomy to categorize TML algorithms. Several books [39], [41], and [1], survey articles [40], and research papers address the various algorithms. Goldstein and Uchida [41] provide a comprehensive review of nineteen different unsupervised TML algorithms with ten multivariate datasets and provide computer runtime estimates for each algorithm.

Suitability for streaming is dependent on the efficiency of the algorithm. Algorithmic efficiency is a property that captures the computational resource requirements as a function of the size of the input *n*. The most common notation is the 'Big $\mathcal{O}$' notation. $\mathcal{O}(1)$ indicates constant time with respect to n; $\mathcal{O}(\log n)$ means logarithmic with respect to n; and $\mathcal{O}(n)$ means linear with respect to n. Many anomaly detection algorithms are $\mathcal{O}(n^2)$, quadratic and $\mathcal{O}(c^n)$, exponential, where c>1, which may be too time-intensive to be suitable for stream processing. Unsupervised algorithms that could be adopted and operate at $\mathcal{O}(n)$ or marginally slower are potentially suitable for use with streaming data.

With univariate temporal data, autoregressive integrated moving average (ARIMA) models and all of their variations have been used to predict point anomalies in unsupervised and semi-supervised univariate time-series data [42]. ARIMA models are particularly popular because of their ability to smooth moving averages to eliminate noise and the inclusion of terms that expresses drift, noise, and non-stationarity over time. Point anomalies in consecutive data points are easily identified.

Anomaly identification is more difficult with multivariate data. Several TML techniques, such as multivariate regression, principal components analysis (PCA), and other linear models,

are often used with varying degrees of success to identify anomalies in multivariate data. These easy-to-implement linear modeling techniques, while useful for predicting a few steps or large deviations, are less useful when past data demonstrate unusual shapes or patterns. More sophisticated non-linear techniques such as DNNs are required. For example, in autonomous driving applications, large fluctuations in the shape of sensor information are expected depending on the road conditions, terrain, and weather. Collective anomaly detection techniques are required with temporal data collected over the entire operational history.

TML-based temporal anomaly detection studies include [43], [44] and [45]. Salechi and Rashidi [46] provide a survey of anomaly techniques in the presence of evolving or changing data. Some studies have specifically incorporated spatiotemporal relationships into their models, such as dynamic environmental monitoring campaigns [47]. Discrete event anomaly detection methods are described in [48]. Other TML-based anomaly detection and related studies include [49], [50], [51], [52] and [53].

4.2 Experimentation Algorithms

Algorithms here are allocated to four (4) categories: Linear Models, Proximity Models, Ensemble Techniques, and Statistical Models. Table 9 lists eight (8) selected TML-based techniques that could be adapted to support streaming anomaly detection. The list includes the most popular and mature unsupervised techniques and excludes supervised techniques. Some of these techniques were designed explicitly for anomaly detection (e.g., Local Outlier Factor) whiles others are generic but adaptable (e.g., Principal Components Analysis). Algorithms with the term ‘local’ in their name are proximity-based and are designed to detect local outliers. The experimentation datasets described in Chapter 3 contain global anomalies. For this reason, local

techniques will perform less optimally on these datasets. Each selected algorithm is briefly discussed below.

Table 9: Traditional Machine Learning Unsupervised Anomaly Detection Techniques

TML Category	Algorithm	Local/Global	Reference
Linear Models	One Class Support Vector Machine (OC-SVM)	Global	Scholkoft, Platt, Shawe-Taylor, Smola, and Williamson [54].
	Principal Component Analysis (PCA)	Global	Aggarwal [55].
Proximity Models	K Nearest Neighbor (K-NN)	Global	Ramaswamy, Rastogi, and Shim [56].
	Local Outlier Factor (LOF)	Local	Breunig, Kreigel, Ng, and Sander [57].
	Cluster-Based Local Outlier Factor (CB-LOF)	Local	He, Xu, and Deng [58].
	Histogram-Based Outlier Score (HBOS)	Global	Goldstein and Dengel [59].
Ensemble Techniques	Isolation Forest (IF)	Global	Liu, Ting, and Zhou [60].
Probabilistic Models	Minimum Covariance Determinant (MCD)	Global	Hardin and Rocke [61], Rousseeux and Driessen [62].

4.3 One-Class Support Vector Machine (Linear Model)

Support Vector Machines (SVMs), a popular supervised TML technique, was modified by Scholkoft, et al. [54] to become a semi-supervised and unsupervised technique known as the One-Class Support Vector Machine (OC-SVM). An SVM implicitly maps the data to a high-dimensional space and separates classes using a linear classifier. OC-SVM is similar except that the algorithm attempts to separate the instances in high dimensional space from the origin. The OC-SVM assumes that the training data is free of anomalies, drawing a boundary in what is known as kernel space around the normal class. OC-SVM is trained using the training data. Test samples are scored using a normalized distance to a decision boundary. Samples that do not fall

within the threshold distance are presumed to be anomalous. Borrowing from the operations research community, OC-SVM can be formulated as a quadratic programming minimization problem using Lagrange techniques and, as most optimization problems, is compute-intensive, and does not scale linearly. For this reason, OC-SVM is an unlikely candidate for streaming applications. [63] provides a good description of OC-SVM in the context of anomaly detection. Ma and Perkins [64] illustrate an example of the use of OC-SVM for temporal anomaly detection.

4.4 Principal Components Analysis (Linear Model)

Principal Components Analysis (PCA) has a long history in statistical science dating back to the early 1900s as a technique for dimensionality and covariance reduction. PCA is a fast, low-overhead procedure that converts a set of correlated, multivariate vectors into a smaller set of linearly independent, orthogonal vectors. The DNN analogy to PCA is autoencoders; the difference is that PCA is a linear combination of vectors using linear algebra techniques while an autoencoder is a nonlinear combination of vectors using DNN techniques. Both PCA and autoencoders are two way; vectors can be encoded and decoded with limited loss of information. The basic intuition is that anomalies can be identified through reconstructive errors. If for a given sample, the original set of input values are not closely replicated after the decoding process, an anomaly is indicated. Dimensionality reduction and DNN autoencoders are discussed in more detail in Chapters 6. See Hinton and Salakhutdinov [65] regarding the benefits of dimensionality reduction.

4.5 K-Nearest Neighbor (Proximity-Based Model)

The K Nearest Neighbor (K-NN) is a proximity-based technique that considers the distance between adjacent samples with algorithms to spatially group samples. While the

techniques are simplistic with two-dimensional data, the computational complexity increases exponentially with multivariate data. Various distance measures used to calculate the k-NN is described in Upadhyaya and Singh [66]. Tsai and Lin [67] provide an application of k-NN to intrusion detection.

For every sample in the dataset, the k-nearest neighbors are identified based on the distance measure. An anomaly score is calculated using these neighbors with either two approaches. One approach is to use the distance to a single k^{th} nearest neighbor, known as the k^{th}-NN technique, and the second approach is to use the average distance over all neighbors, known as the k-NN technique. The average distance may either be the mean distance or the median distance across all neighbors. The k-NN technique is substantially more computationally expensive than the k^{th}-NN technique. With both techniques, the choice of the k parameter is critical, typically $10 < k < 50$. Note that the nearest neighbor and other proximity-based algorithms operate at $\mathcal{O}(n^2)$.

4.6 Local Outlier Factor (Proximity-Based Model)

The Local Outlier Factor (LOF) algorithm compares the density of samples around a given high-dimensional region. The LOF algorithm compares the density of instances around a given instance to the density of the neighbors. The LOF approach is similar to the k-NN technique but applied within a localized data area. The algorithm defines an average local reachability density or LRD. LOF is then calculated by dividing the average LRD of all localized neighbors by their LRD. A LOF ~ 1 means that the data point has a similar density of the neighbors; a LOF < 1 signifies higher density than the neighbors indicating a normal sample and a LOF > 1 signifies lower density than the neighbors indicating an anomaly.

4.7 Cluster-Based Local Outlier Factor (Proximity-Based Model)

There are many different varieties of clustering; one popular technique designed for anomaly detection is Cluster-Based Local Outlier Factor (CBLOF). The underlying assumption is that normal samples belong to that cluster closest to the cluster centroid, while anomalous samples do not belong to any cluster. CBLOF is similar to LOF but classifies the data into a set of small clusters and large clusters. The algorithm calculates an anomaly score based not only on the size of the cluster but also on the distance to the nearest larger cluster. For a complete discussion of CBLOF, see (He, Xu, and Deng, [58]). In general, clustering algorithms operate faster ($<\mathcal{O}(n^2)$) than nearest neighbor techniques.

4.8 Histogram-Based Outlier Score (Proximity-Based Model)

The Histogram-Based Outlier Score (HBOS) is a nonparametric technique designed to be extremely fast in execution. The underlying assumption is that each feature in the dataset is independent, and an anomaly score can be calculated by building a set of histograms. While the assumption of independence may result in reduced accuracy, the speed of computation makes HBOS a possible candidate as a streaming algorithm. HBOS also has the concept of dynamic bins where the number of histogram bins remains constant, but the size of the bins increases or decreases as needed. The advantage of this dynamic approach is that density estimation is exceptionally flexible in the presence of a large number of anomalies. Equation (4.1) provides the scoring algorithm for the HBOS technique; the algorithm calculates the sum of probabilities over the entire histogram bins $i = 1 \ldots d$:

$$HBOS\ Score = \sum_{i=i}^{d} \log\left(\frac{1}{hist_i(p)}\right) \tag{4.1}$$

Note that HBOS is a global, non-parametric technique, operates at $\mathcal{O}$(n), and is a candidate for stream processing adoption.

4.9 Isolation Forest (Ensemble Technique)

Ensemble techniques combine two different techniques into a single technique. An Isolation Forest (IF) is an example of an anomaly detection ensemble technique. The IF approach is dramatically different from the one-class SVM approach and uses decision trees to identify anomalies. First, an attribute is selected, and a partition is created by randomly selecting a value between the minimum and maximum value of that attribute. Since anomalies are, by definition, scarce and distant from normal values, a normal sample will require more partitions than an abnormal sample. Based on IF technique, an anomaly score is given by:

$$s(x,n) = 2^{-\frac{E(h(x)}{c(n)}} \tag{4.2}$$

where h(x) is the path length of sample x, c(n) is the average path length of an unsuccessful search in a Binary Search Tree, and n is the number of external nodes. Based on this formulation, anomaly scores close to one indicate the high possibility of an anomaly, while a score closer to .5 would indicate that an anomaly is unlikely.

In summary, an IF uses binary trees and can be scaled up to process large, high-dimensional datasets. Anomalous samples are usually far (in high-dimensional space) from other samples, so on average, across all the decision trees, anomalies are isolated and identified in fewer steps than normal samples.

4.10 Minimum Covariance Determinant (Probabilistic Model)

Minimum Covariance Determinant (MCD) is a density-based approach based on Mahalanobis distance that has been applied to anomaly detection problems (see Section 4.10.1 below for an aside on Mahalanobis distance). With probabilistic models of anomaly detection,

each sample is analyzed against a probability distribution; the anomaly score is the probability of occurrence. This approach to anomaly detection is called the Gaussian technique because of the underlying assumption of the data is normality. However, the density model approach can be applied to other statistical distributions such as Gamma or Chi-Squared distributions. When the underlying distribution of the non-anomalous population is unknown, which is often the case, a mixture of Gaussians is used as a proxy for the correct distribution.

Consider a random variable $X = [X_1 \ldots X_n]$ with a mean $\mu \in \mathbf{R}^n$ and covariance matrix Σ^{nxn}, (positive definite), the standard Gaussian probability density function is given by equation (4.3):

$$p(x, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^{T}\Sigma^{-1}(x-\mu)\right) \tag{4.3}$$

Where $|\Sigma|$ denotes the determinant of the covariance matrix.

To fit the parameters, given a semi-supervised training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, estimate $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{i}$ and $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^{T}$, then for the new (or test) samples, fit model p(x) by setting μ and Σ, compute p(x) using equation (4.3) and flag as an anomaly if $p(x) < \in$ for some threshold value of $\in$.

The MCD approach assumes that the non-anomalous samples are generated from a single Gaussian distributed and not a mixture of Gaussians. When the algorithm estimates the parameters of the Gaussian distribution by estimating the shape of the elliptic envelope while excluding the anomalous samples so as not to distort the parameter estimates. Note that the Gaussian density estimation approach is most useful for unsupervised point anomaly detection but would be intractable and difficult to implement for lengthy temporal streaming data. Moreover, there is no *a priori* basis to determine the correct sequence length for parameter

estimation so all sequence length combinations would need to be tested. While online streaming processing is feasible, this model can be computationally expensive because the calculation Σ^{-1} is a matrix inversion, which can be costly for a large number of features. Density estimation may be too slow for streaming data since the performance is $\mathcal{O}$(n) or slower. Moreover, the training set size must be larger than the number of features for Σ to be invertible.

4.10.1 An Aside on Mahalanobis Distance

The Mahalanobis distance is a crucial distance measure used in TML algorithms, including MCD. From equation (4.3) above, which is the equation of a Gaussian distribution, the term $(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu))$ is one-half of the squared Mahalanobis distance from the of the data point $x^{(i)}$ and the mean μ of the data. The Mahalanobis distance d is the anomaly detection score; the computation is shown in equation (4.4):

$$Mahalanobis(x,\mu,\Sigma) = \mathrm{d} = \sqrt{(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)} \tag{4.4}$$

Note that this distance method is parameter-free, and although quadratic in terms of data dimensionality, the measures are linear in the number of data points, an essential consideration for computational efficiency when processing streaming data. For normal multivariate distributions, the probability density of the Mahalanobis distance is chi-squared distributed.

For streaming data, a simplified version of Mahalanobis distance shown in equation (4.5) is easy to implement, where *x* is a vector containing all of the dimensions of a single sample; μ is a vector representing the mean or center of mass of all of the data samples, and n is the number of elements in x; and μ and $d(x^i,u^i)$ is the difference between the i^{th} element of x and μ:

$$Streaming\ Mahalanobis(x,\mu) = \mathrm{d_s} = \sum_{i=1}^{n} \frac{d(x^i,\mu)}{\sigma^i} \tag{4.5}$$

4.11 Summary

Eight (8) TML anomaly detection algorithms are described in this chapter. These algorithms are unsupervised designed to address point anomalies for non-streaming data. However, since many of these algorithms are not compute-intensive, most can be adapted for streaming data. Some of these algorithms can execute in a few seconds with gigabyte datasets. Even if the application requires large datasets or the application or use case timing requirements are in the milliseconds rather than seconds, a moving window can be employed to detect anomalies on streaming subsets of the data using the same techniques described here. Moreover, because these are TML unsupervised techniques, there is no requirement to pre-train the models before their use in streaming applications.

TML algorithms, however, process one sample at a time and cannot model complex relationships that exist over long periods or across spatial regions. In short, the TML algorithms cannot robustly identify contextual or collective anomalies. Chapter 6 discusses the DNN approach to anomaly detection that could model complex inter-relationships with long-term memory to identify contextual or collective anomalies in spatiotemporal data. However, there is a trade-off; DNN algorithms are processor-intensive to the extreme and may require minutes to hours to train. DNNs can be pre-trained if there was a belief that the model parameters exhibit long-term stability over space and time. However, in many domains, there is a requirement to process streaming data not only for near real-time processing but also to identify concept drift in the underlying relationships in the data.

The next chapter describes experimentation with the TML algorithms described in this chapter. The goal of this experimentation is to evaluate algorithmic performance and to determine if these algorithms can be adapted for streaming applications. These results and

conclusions will provide a baseline to compare against the findings associated with the DNN anomaly detection experimentation that are discussed in Chapters 6 and 7.

CHAPTER 5 – TRADITIONAL MACHINE LEARNING (TML) EXPERIMENTATION

5.1 Experimentation Overview

The eight (8) TML algorithms described in Chapter 4 are the subject of experimentation and evaluation. These algorithms include Cluster-Based Local Outlier Factor (CB-LOF), Histogram-Based Outlier Score (HBOS), Isolation Forest (IF), K-Nearest Neighbors (K-NN), Local Outlier Factor (LOF), Minimum Covariance Determinant (MCD), On-Class Support Vector Machines (OC-SVM), and Principal Component Analysis (PCA). Three (3) variants of the k-NN algorithm are evaluated, including point-to-point, mean, and median nearest neighbors. Therefore, in total, ten (10) algorithms are compared for execution time and test performance on each of the four labeled anomaly datasets discussed in Chapter 3.

Each dataset was separated into a training sample (70 percent) and a test sample (30 percent). The training dataset was used for parameter estimation, and the test dataset was used for performance evaluation. All algorithms are unsupervised, meaning that the anomaly labels are not used for parameter estimation. Anomaly labels are used for test and evaluation purposes. The performance evaluation criteria are described in Section 2.5. Key metrics include algorithm execution time, Area under the Recovery Operating Characteristic (ROC) curve (AUC), Precision, Recall, and the F_1 metric. Execution time is essential when evaluating the suitability of an algorithm for adaptation for stream processing. All experimentation was performed on the identical hardware that included a NVIDIA™ 2060 Graphical Processing Unit (GPU). Further information on the software estimation approaches and the software packages can be found at [68].

5.2 Experimentation Execution Time

Table 10 presents the wall-clock execution times in seconds for each of the algorithms with each experimentation dataset. The results across algorithms are comparable for a given dataset only. Each dataset has unique characteristics, including the number of features, the number of samples, and the number of labeled anomalies that would make comparisons across datasets problematic. Processing time is dependent on the number of features included in the analysis. The scalability of the algorithm is dependent on the performance characteristics, and, of course, the efficiency of the particular software implementation. Every algorithm exhibits the same or longer processing time when all features are processed compared to a subset of the dataset features, but the increase in processing is non-linear. The FRAUD dataset included thirty (30) features with 227846 training samples; the SWAT dataset included forty-six (46) features with 496800 training samples; the WADI dataset included ninety-nine (99) features with 1048571 training samples, and the DATACENTER dataset included seven (7) features with 134400 training samples.

There are significant differences among the algorithms in terms of execution speed. PCA and HBOS exhibited the fastest execution times and are the most linearly scalable. PCA is based on linear algebra, while HBOS is non-parametric, involving histograms and data binning. PCA is extremely fast on smaller datasets but has difficulty scaling to more massive datasets. The fact that HBOS was the highest performant algorithm is not unexpected; the technique has performance characteristics of $\mathcal{O}$(n). HBOS processed the most stringent test, the WADI dataset, in 5.1 seconds. Other algorithms, such as CB-LOF, performed quickly with the smaller DATACENTER dataset (less than one second) but did not linearly scale to the WADI dataset (203 seconds).

` The three (3) k-NN algorithms were not particularly scalable, which is consistent with the expected run-time performance characteristic of $\mathcal{O}(n^2)$. The k-NN processed the DATACENTER dataset in about thirty (30) seconds, jumped to 1363 seconds for processing the FRAUD dataset, and then to 15568 seconds for the WADI dataset.

The OC-SVM algorithm failed to complete processing WADI, the largest dataset. SVMs are known to be processor intensive and non-scalable, so this result is expected. Other proximity-based (k-NN, LOF) and probabilistic algorithms (MCD) demonstrated long execution times and are not suitable for adaptation for stream processing.

Table 10: Traditional Machine Learning Clock Times (Seconds)

Technique	FRAUD (30)	SWAT (46)	WADI (99)	DATACENTER (7)
Training Samples	227846	496800	1048571	134400
CB-LOF	19.1	22.4	203.8	.78
HBOS	1.5	1.9	5.1	1.2
I-FOREST	26.7	66.8	294.6	7.5
k-NN	1363.8	1405.7	15568.5	29.9
k-NN (Mean)	1697.7	1782.3	16431.4	50.8
k-NN (Median)	1921.6	1453.4	16413.0	75.0
LOF	2258.9	1500.3	2613.1	68.8
MCD	66.7	223.8	2025.1	14.2
OC-SVM	3313.9	21648.7	DNF – DID NOT FINISH	754.9
PCA	0.7	3.1	29.7	0.1

5.3 Experimentation Performance Results

Algorithmic speed is of little value if the results produced are of low accuracy or high variance. Each algorithm was evaluated against each experimentation dataset. Anomaly datasets are typically unbalanced, meaning only a small percentage of all samples are anomalous. As previously noted, in order to compare performance across algorithms and datasets, the area under the Receiver Operating Characteristic (ROC) curve (AUC) is the preferred metric. Recall that the higher the value of AUC, the higher the accuracy of the algorithm; estimates of AUC of .8 and above are generally viewed as favorable.

The AUC, Precision, Recall, and F_1 metrics are based on the parameters estimated on the training data applied to the test dataset. Two approaches to the creation of the test dataset are possible. The first approach is to separate anomalous samples from the training dataset and allocate these samples to the test set. This approach is possible because the anomalies are labeled; however, in real-world situations, anomalies will not be labeled, and the anomaly rate is low. The second approach is to separate the dataset into training and test subsets without regard to the anomaly labels. This latter approach is more operationally realistic and was adopted for use with the FRAUD and DATACENTER datasets. However, since the SWAT and WADI datasets were created from testbeds, there is a natural division between the training and test datasets. The training datasets are based on the operation of the testbeds under normal conditions. In contrast, the test datasets are based on the operation of the testbed during the period of attack.

The experimentation findings are presented in Tables 11-14. Table 11 presents the results for the FRAUD dataset; Table 12 for the SWAT dataset; Table 13 for the WADI dataset; and Table 14 for the DATACENTER dataset. The tables also include estimated predictions for true-negatives, false-negatives, true-positive, and true-negative, which are the numbers that are entered into the 'confusion matrix' described in Section 2.5.1. Note that an anomaly is considered a positive, while a non-anomaly is negative.

Each algorithm defines a particular methodology for anomaly scoring. By convention, higher scores indicate high probabilities of anomalies. However, anomaly scores are not comparable across algorithms; only the rank order is essential. With the test dataset, each sample is scored (based on the parameters estimated from the training dataset) and ranked from high to low. If the population includes 'x%' anomalies, then the top 'x%' of the anomaly scores in the

test dataset are designated as anomalies and are compared against the actual labels. The values of a confusion matrix can then be calculated.

Note that a confusion matrix is based on a single threshold value of an anomaly score, while the AUC metric is derived from all possible threshold values. Note that the values for Precision, Recall, and F_1 metrics are for the single-valued, algorithmically defined scoring threshold, not a range in scoring values. Ranges in Precision and Recall can be displayed graphically as a function of the algorithm threshold. Figures 16-24 provides three sets of graphs associated with each dataset - algorithm combination: (a) the precision/recall vs. threshold graph, (b) the precision vs. recall graph, and (c) the Receiver Operating Characteristic (ROC) curve.

5.3.1 FRAUD Experimentation

Table 11 displays the findings with respect to the FRAUD dataset. All algorithms produce an AUC above .9 except for Local Outlier Factor (LOF), where the AUC=.505. The likely explanation for this result is that the FRAUD data contains only global anomalies, so a local algorithm such as LOF will perform poorly. The fact that all other algorithms produce excellent results indicates that anomalies are easily detected in this dataset, which is often the case with financial anomalies.

Note the example of a confusion matrix produced by the HBOS algorithm. With an AUC=.962 and the algorithm default scoring threshold, there were 85179 true negatives; the algorithm correctly identifies these samples as non-anomalous. HBOS also produced ninety-six (96) false negatives, anomalies that were not identified; HBOS produced 107 false positives, non-anomalies that were incorrectly identified as being anomalous. Finally, there were sixty-one (61) true positives, anomalies that were correctly identified.

5.3.2 SWAT Experimentation

Table 12 presents the AUC findings for the SWAT dataset, which is similar to the findings for the FRAUD dataset. All algorithms produce AUCs that exceed .84, except for LOF (AUC=.793). Note that, for example, at the KNN default scoring threshold, no true negatives and no false negatives are identified. All true-positives and all false-positives were identified. This result means that the algorithm threshold defaults are outside the relevant range and incorrectly set.

5.3.3 WADI Experimentation

Table 13 presents the findings associated with the WADI dataset. All AUC measurements are below .8 except for the PCA algorithm (AUC=.82). WADI is the largest dataset with the most features and samples. This result might indicate that the quality of the WADI is lower than the other datasets, that the anomalies introduced into the testbed were not actual anomalies, or that the algorithms are insufficiently robust to uncover the complex interactions of the dataset features.

5.3.4 DATACENTER Experimentation

Table 14 presents the findings associated with the DATACENTER dataset. Three algorithms (k-NN (Mean), k-NN (Median), and LOF) recorded an AUC > .8. LOF correctly identified 128 anomalies in the data. This result might indicate that the data includes local rather than global anomalies. This dataset is a pooling of the samples from one-hundred (100) different datacenters. By pooling the data across datacenter, the possibility exists that local, datacenter-unique anomalies predominate.

Table 11: FRAUD Experimentation Results

		Confusion Matrix						
Algorithm	**AUC**	**True Negative**	**False Positive**	**False Negative**	**True Positive**	**Precision**	**Recall**	**F_1**
CB-LOF	.964	85166	120	112	45	.272	.286	.279
HBOS	.962	85179	107	96	61	.363	.388	.375
I-Forest	.952	85175	111	105	52	.319	.331	.325
KNN	.954	85149	137	139	18	.116	.114	.115
KNN-Mean	.942	85140	146	138	19	.115	.121	.118
KNN-Median	.916	85148	138	140	17	.109	.108	.108
LOF	.505	85127	159	157	0	.000	.000	.000
MCD	.960	85252	34	29	128	.790	.815	.802
OC-SVM	.958	85199	177	133	24	.119	.152	.152
PCA	.957	85168	118	116	41	.257	.261	.259

Table 12: Secure Water Treatment Testbed (SWAT) Experimentation Results

		Confusion Matrix						
Algorithm	**AUC**	**True Negative**	**False Positive**	**False Negative**	**True Positive**	**Precision**	**Recall**	**F_1**
CB-LOF	.894	341340	53958	9261	45360	.456	.830	.589
HBOS	.854	293001	102297	11550	43071	.296	.788	.430
I-Forrest	.843	269893	125405	11889	42732	.254	.782	.383
KNN	.908	0	395298	0	54621	.121	1.000	.216
KNN-Mean	.907	0	395298	0	54621	.121	1.000	.216
KNN-Median	.903	0	395298	0	54621	.121	1.000	.216
LOF	.793	111	395187	0	54621	.121	1.0	.216
MCD	.840	344515	50783	14782	39839	.439	.729	.548
OC-SVM	.895	356425	38873	10622	43999	.530	.805	.640
PCA	.891	374400	20898	13249	41372	.664	.757	.707

Table 13: Water Distribution Testbed (WADI) Experimentation Results

		Confusion Matrix						
Algorithm	**AUC**	**True Negative**	**False Positive**	**False Negative**	**True Positive**	**Precision**	**Recall**	**F_1**
CB-LOF	.737	153899	9042	6001	3859	.299	.391	.339
HBOS	.787	158420	4521	6760	3100	.406	.312	.354
I-Forrest	.747	149269	13672	6172	3688	.212	.374	.270
KNN	.776	0	162941	0	9860	.057	1.000	.107
KNN-Mean	.776	0	162941	0	9860	.057	1.000	.107
KNN-Median	.776	0	162941	0	9860	.057	1.000	.107
LOF	.684	4418	158523	118	9742	.057	.988	.109
MCD	.691	147399	15542	6411	3449	.181	.349	.239
OC-SVM	DNF – DID NOT FINISH							
PCA	.821	153706	9235	5993	3867	.291	.391	.333

Table 14: DATACENTER Experimentation Results

		Confusion Matrix						
Algorithm	**AUC**	**True Negative**	**False Positive**	**False Negative**	**True Positive**	**Precision**	**Recall**	**F_1**
CB-LOF	.564	49923	231	245	1	.004	.004	.004
HBOS	.531	49884	270	246	0	.000	.000	.000
I-Forrest	.546	49889	265	245	1	.000	.004	.003
KNN	.776	49926	228	227	19	.076	.077	.077
KNN-Mean	.808	49944	210	219	27	.113	.109	.111
KNN-Median	.801	49935	219	220	26	.106	.105	.105
LOF	.964	50048	106	118	128	.547	.520	.533
MCD	.508	49929	225	246	0	0.0	0.0	0.0
OC-SVM	.546	49932	245	222	245	.004	.004	.004
PCA	.538	49925	229	245	1	.004	.004	.004

5.4 Experimentation Conclusions

Experimentation consisted of ten (10) different unsupervised algorithms trained and tested against four (4) different datasets. The purpose was to determine those algorithms that can produce reasonably accurate results while being adaptable in a streaming environment.

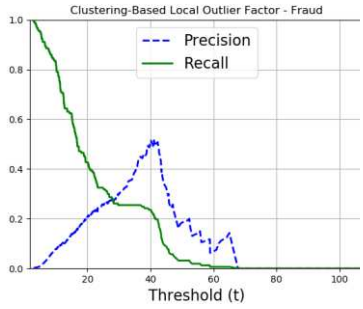
Overall, only the HBOS, I-Forrest, K-NN, and LOF algorithms were able to process all of these benchmark datasets in a reasonable time. HBOS performance was close to the top across all datasets in both the speed and accuracy metrics. The fact that HBOS was near the top is somewhat surprising given the simplicity of the algorithm. HBOS outperformed LOS in all experiments except for the processing of the DATACENTER dataset is also surprising given that LOS is more theoretically justifiable and more popular in the literature than HBOS.

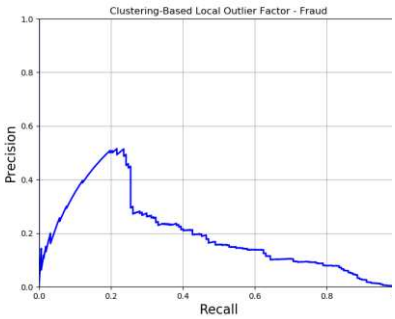
Based on the findings, HBOS appears to be the most promising algorithm for adaption because of the simplicity in design, speed of execution, and reasonably good accuracy vis-a-vis other more complex, theoretically-grounded algorithms. PCA also has demonstrated consistent results. Overall, in terms of anomaly detection accuracy, most algorithms performed similarly on the FRAUD and SWAT datasets and somewhat poorly on the WADI dataset. The results associated with the DATACENTER dataset were mixed, with the LOF algorithm the highest

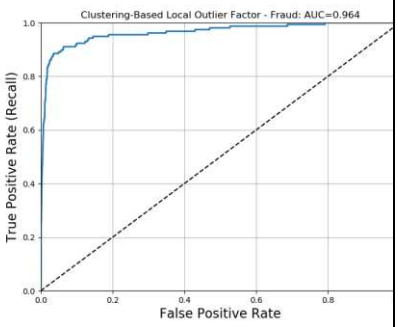
performer due to the likelihood of local anomalies in the raw data. Nevertheless, HBOS, PCA, and other TML techniques used as a singular approach to anomaly detection are unlikely to be adequate for complex multivariate, data-driven spatiotemporal domains.

Chapter 6 discusses the application of DNN models to anomaly detection, while Chapter 7 provides the experimentation results using the same anomaly datasets. The goal is to uncover promising DNN techniques that are adaptable for streaming anomaly detection, and investigate how lightweight TML-based techniques such as HBOS can be integrated into a unified streaming spatiotemporal environment called STADE. The complete STADE specification is presented in Chapter 8.

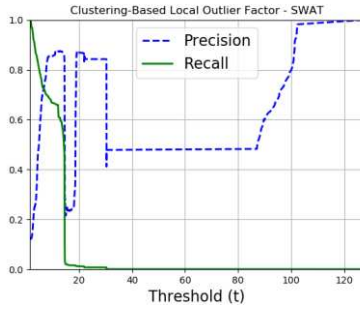
FRAUD

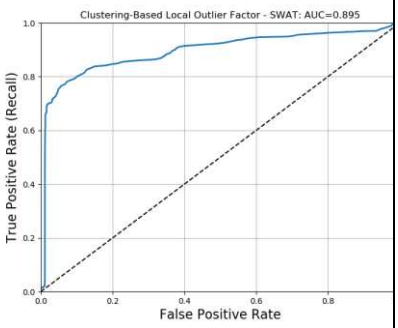




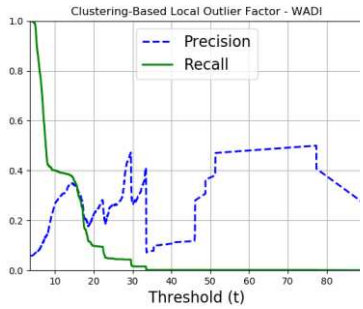


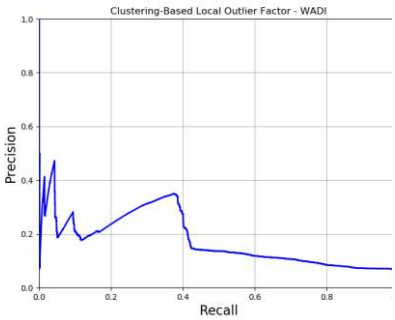
SECURE WATER TREATMENT TESTBED (SWAT)

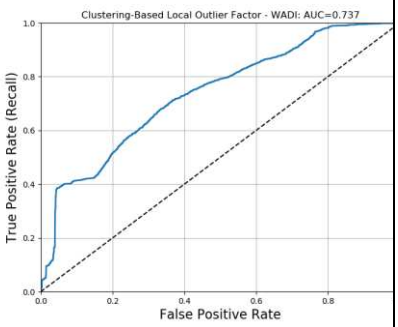




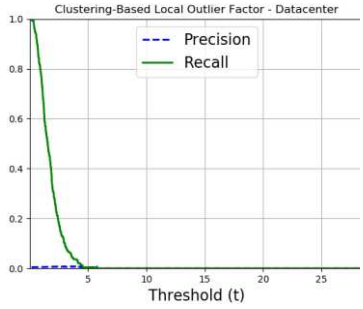
WATER DISTRIBUTION TESTBED (WADI)

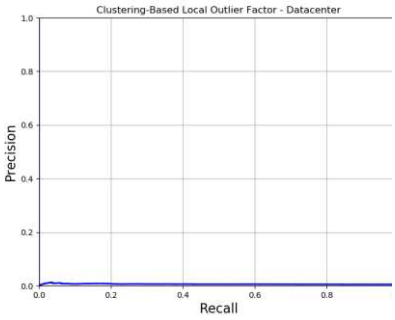






DATACENTER





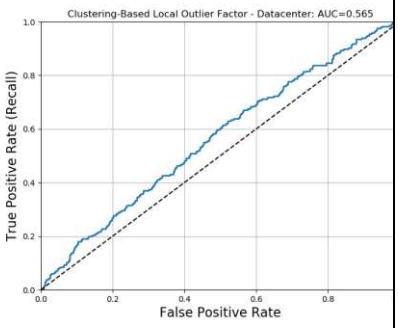
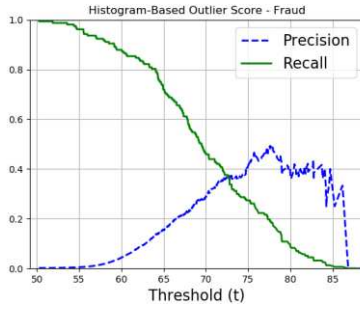
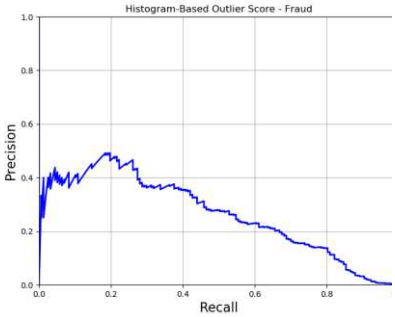
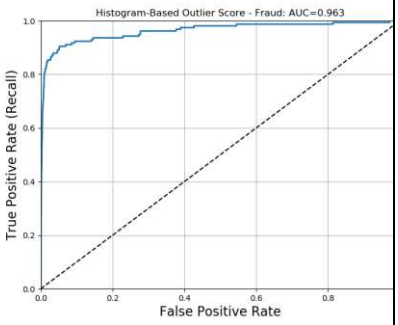


Figure 16 – Clustering-Based Local Outlier Factor (CBLOF)

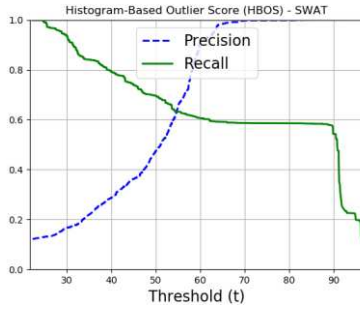
FRAUD

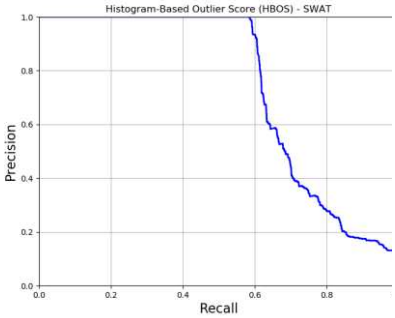




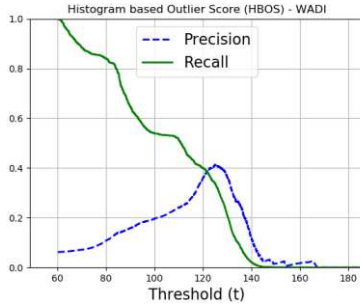


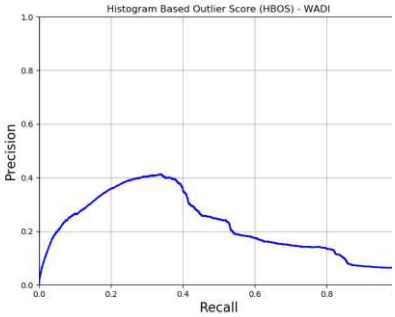
SECURE WATER TREATMENT TESTBED (SWAT)

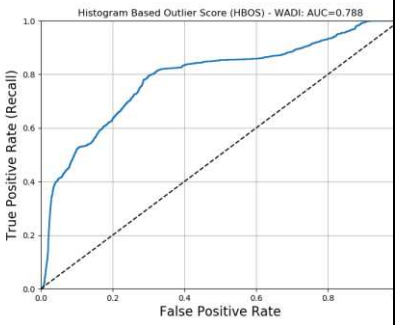




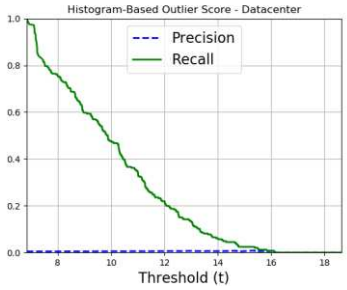
WATER DISTRIBUTION TESTBED (WADI)

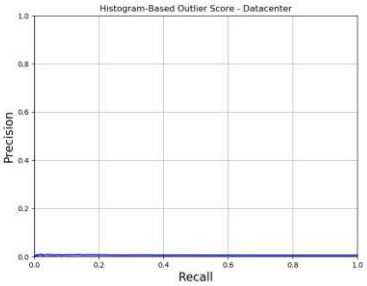






DATACENTER





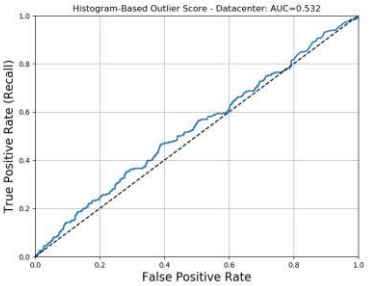
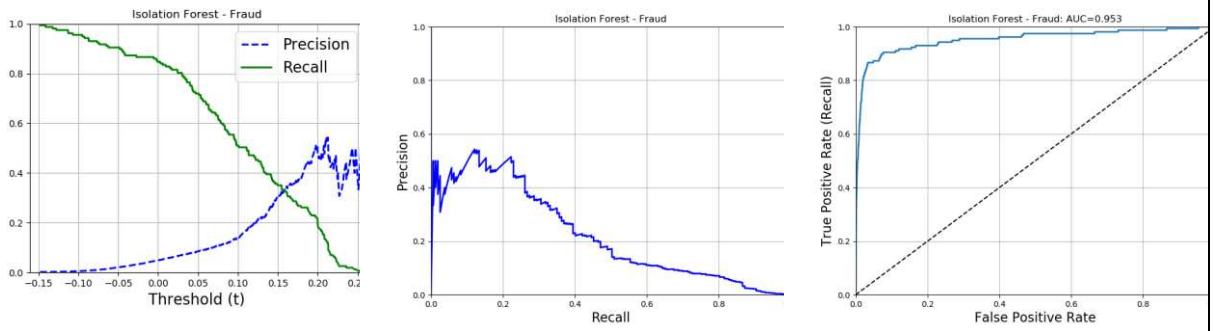
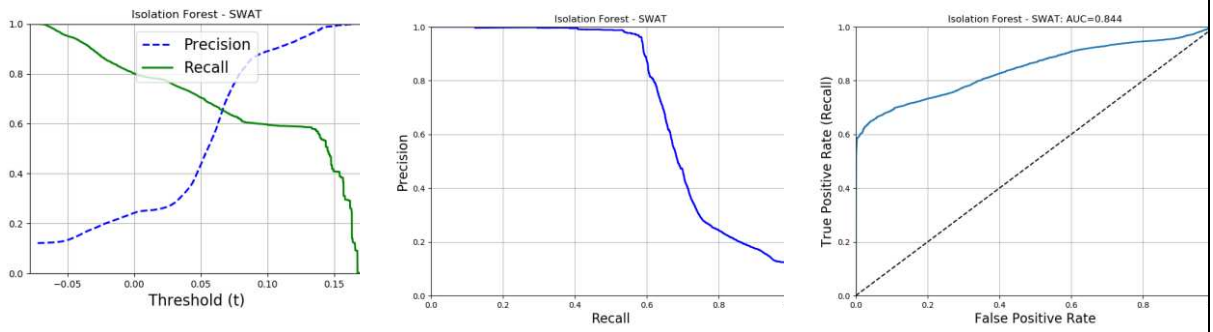


Figure 17: Histogram-Based Outlier Score (HBOS)

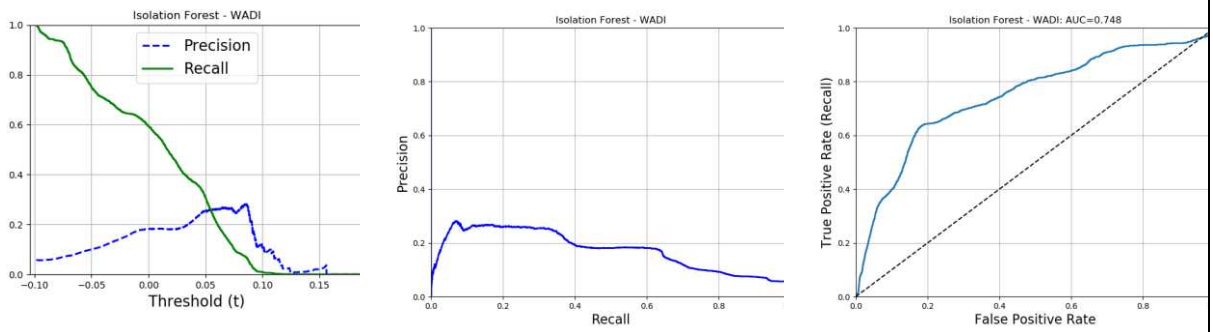
FRAUD



SECURE WATER TREATMENT TESTBED (SWAT)



WATER DISTRIBUTION TESTBED (WADI)



DATACENTER

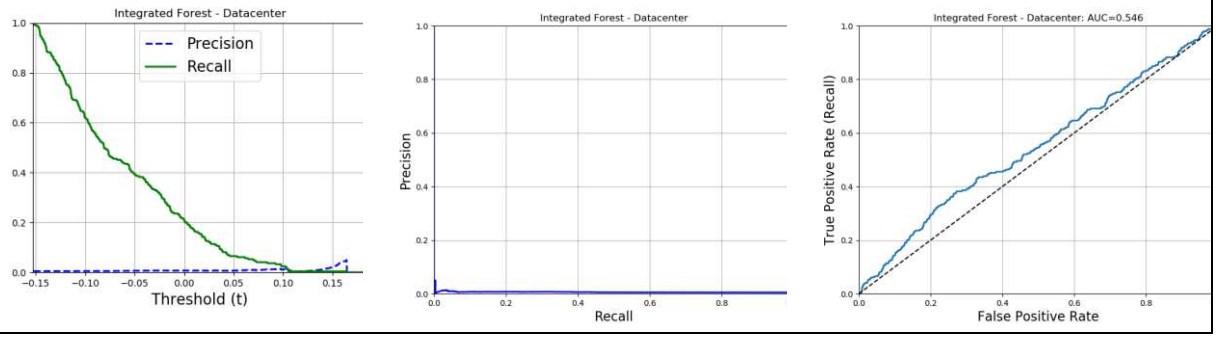
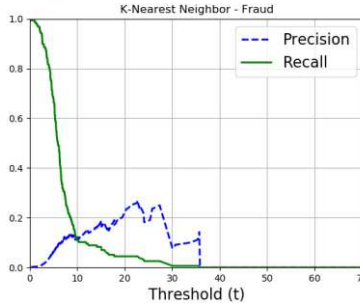
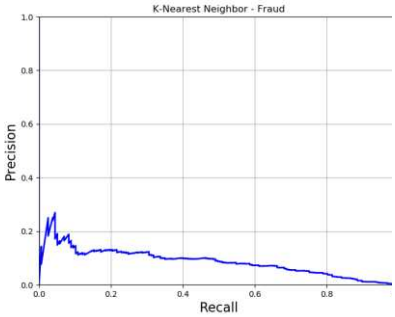
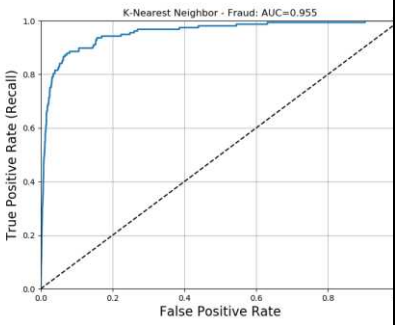


Figure 18: Isolation Forest (IF)

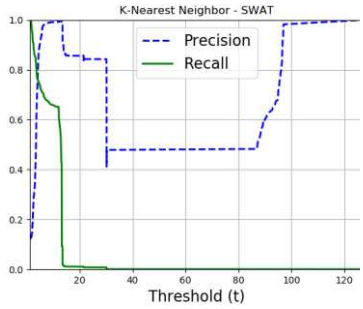
FRAUD

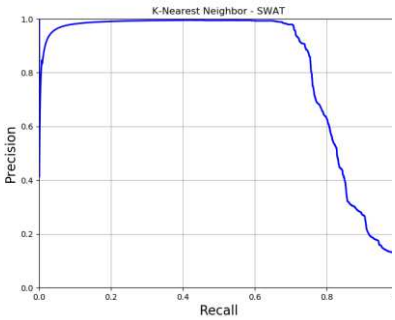


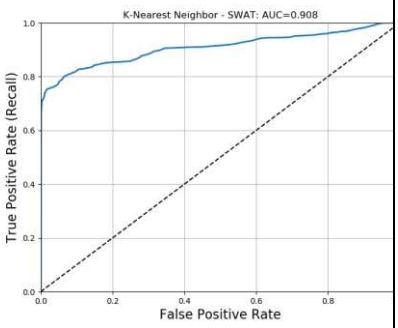




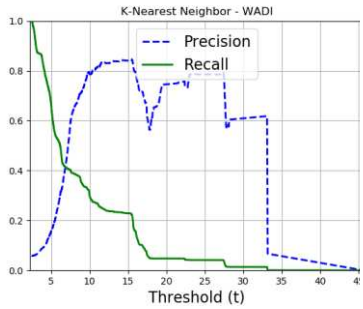
SECURE WATER TREATMENT TESTBED (SWAT)

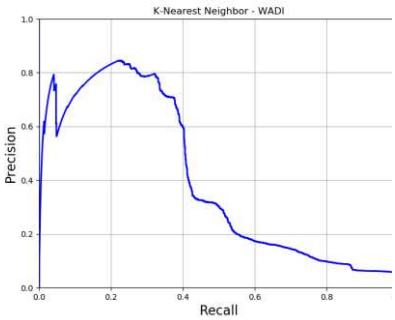


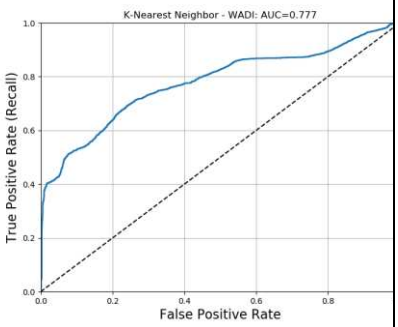




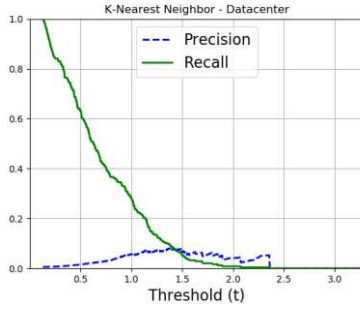
WATER DISTRIBUTION TESTBED (WADI)

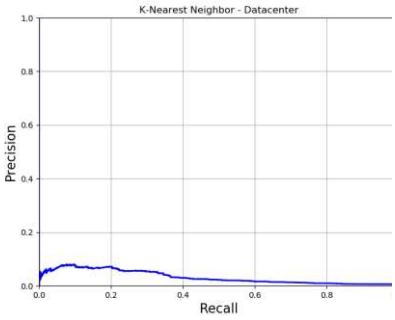






DATACENTER





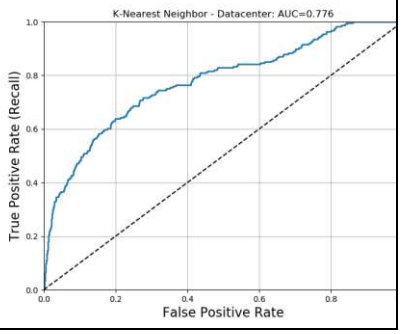
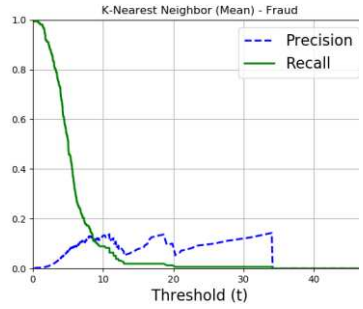
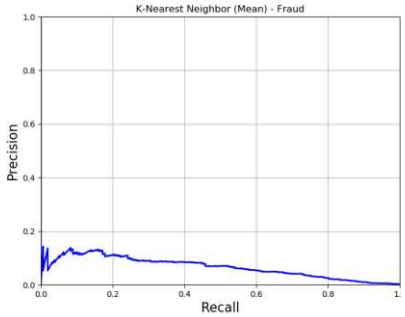
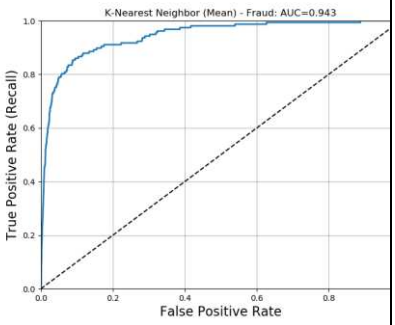


Figure 19: k-Nearest Neighbor (k-NN)

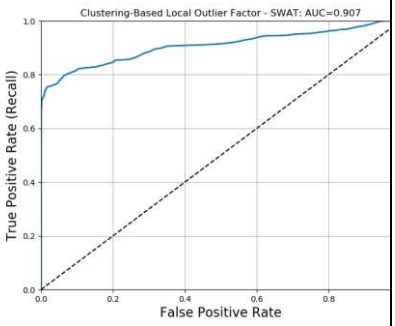
FRAUD



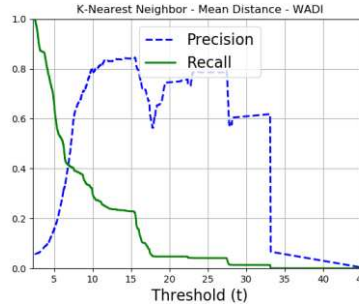


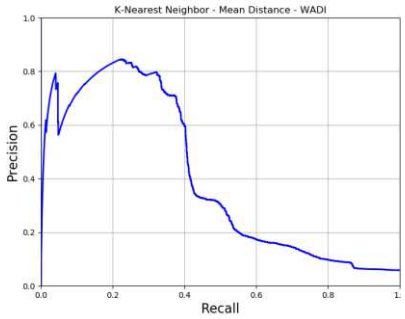


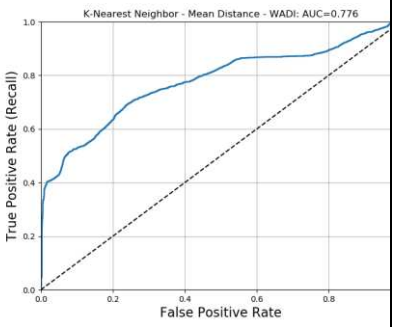
SECURE WATER TREATMENT TESTBED (SWAT)



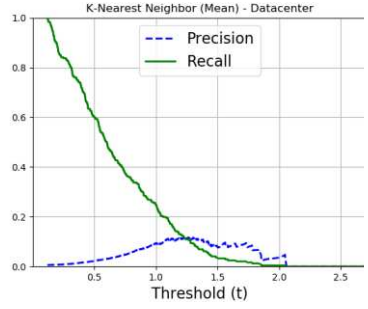
WATER DISTRIBUTION TESTBED (WADI)

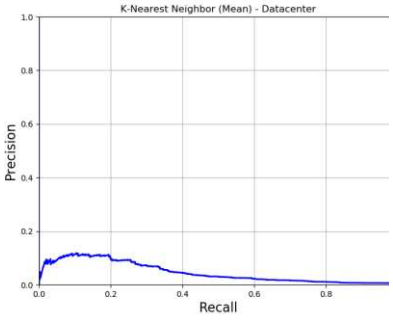






DATACENTER





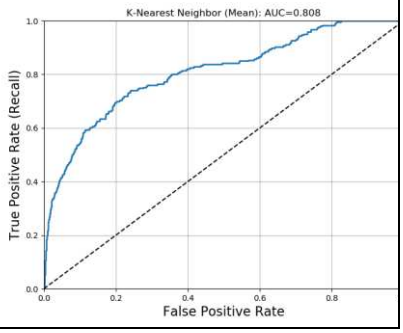
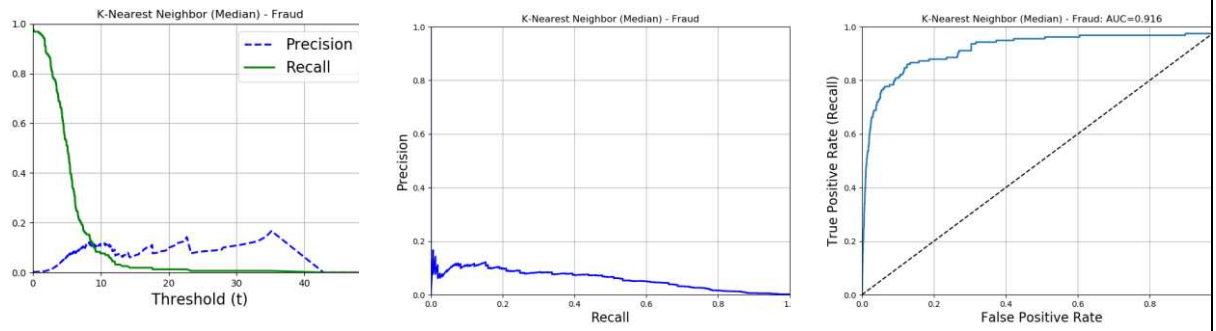
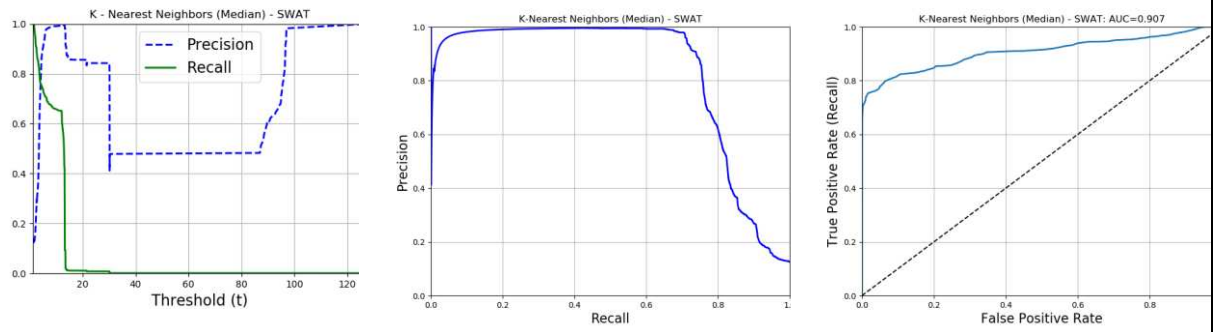


Figure 20: k-Nearest Neighbor (kNN - Mean)

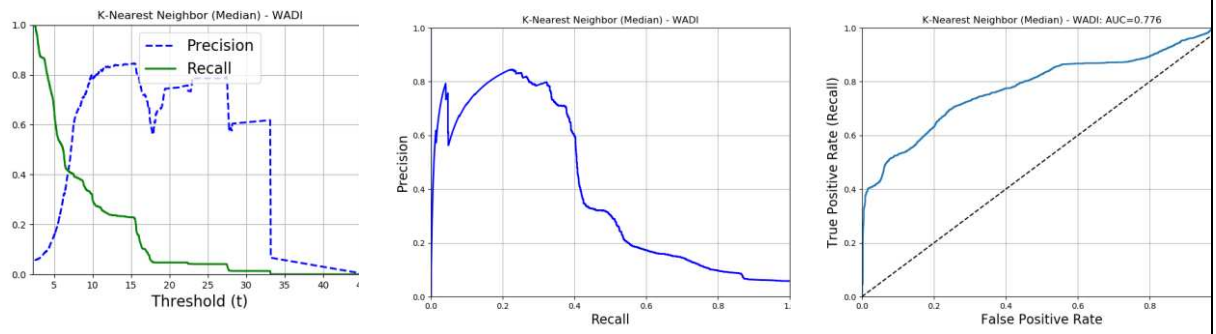
FRAUD



SECURE WATER TREATMENT TESTBED (SWAT)



WATER DISTRIBUTION TESTBED (WADI)



DATACENTER

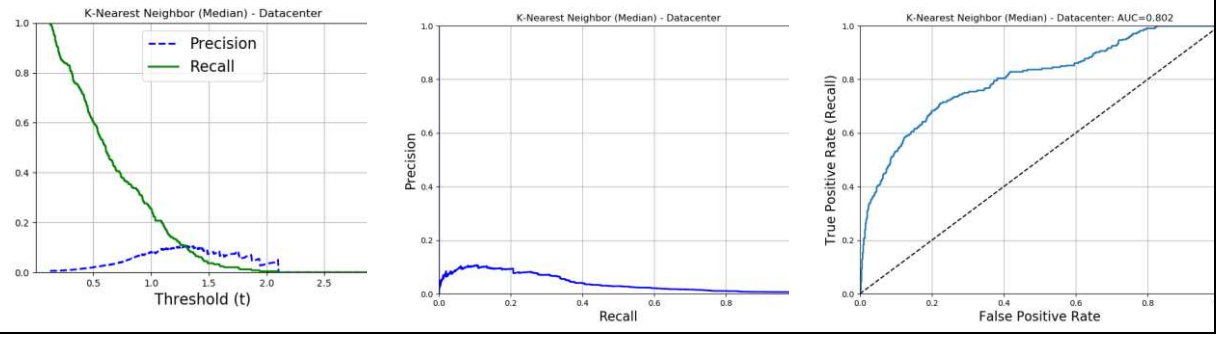
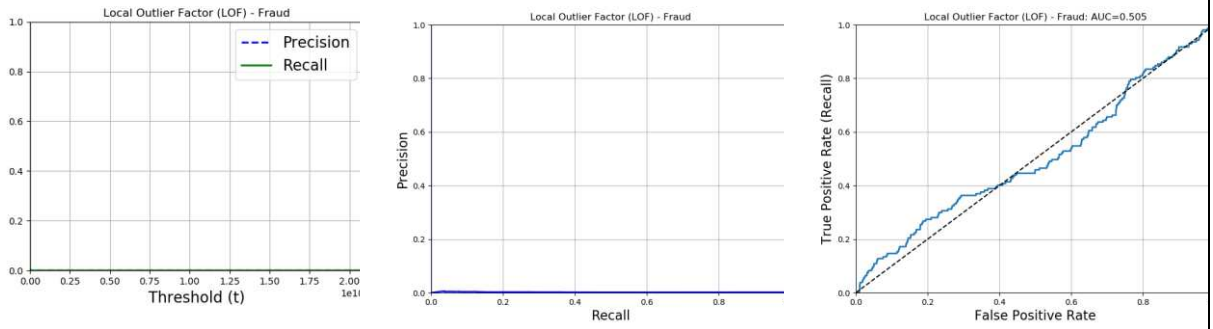
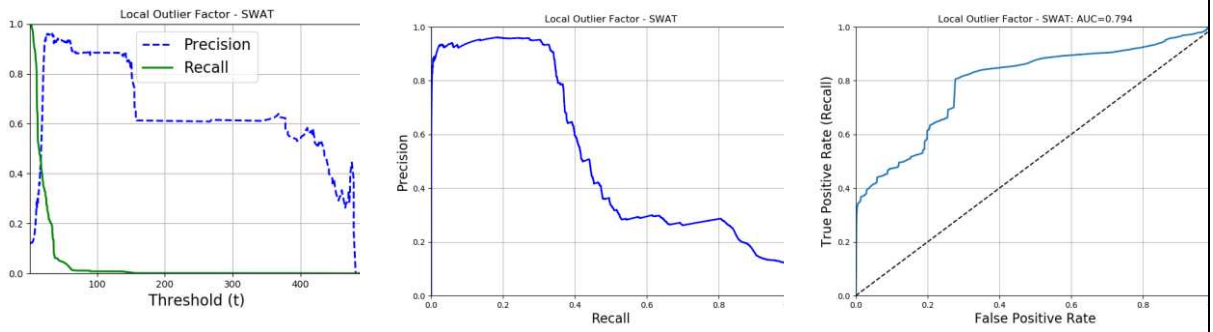


Figure 21: k-Nearest Neighbors (kNN - Median)

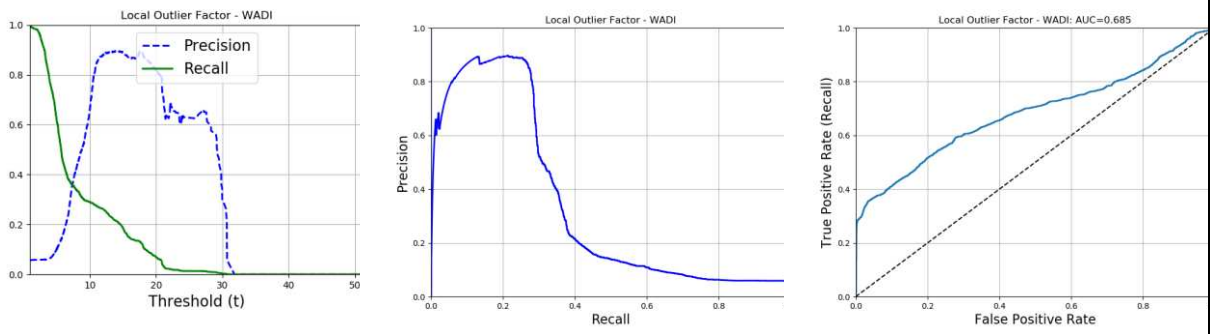
FRAUD



SECURE WATER TREATMENT TESTBED (SWAT)



WATER DISTRIBUTION TESTBED (WADI)



DATACENTER

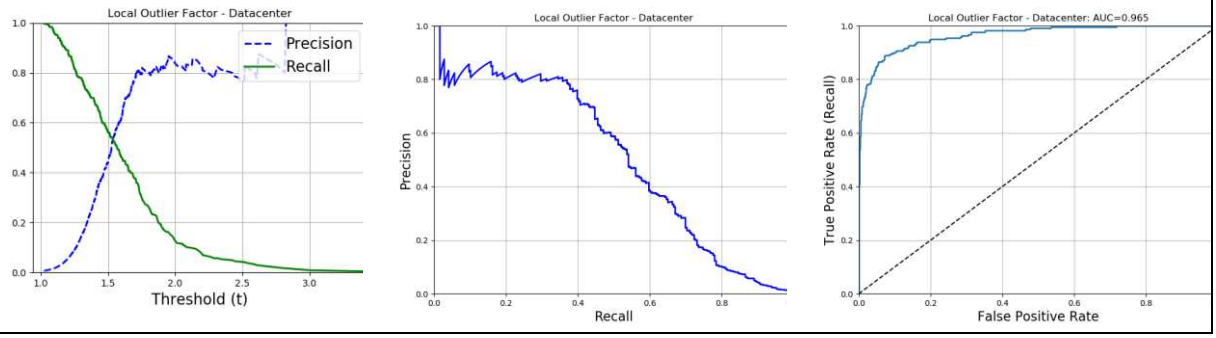
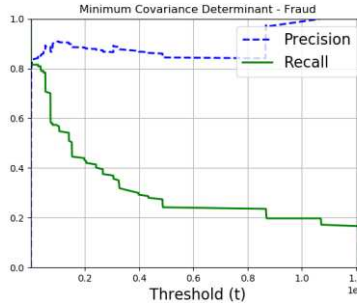
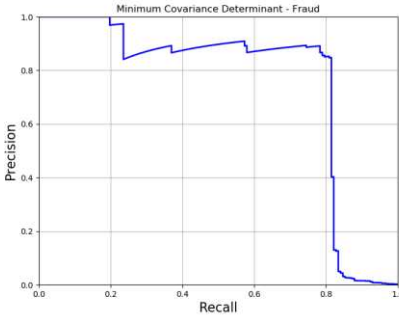
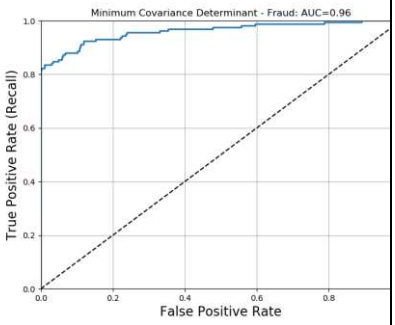


Figure 22: Local Outlier Factor (LOF)

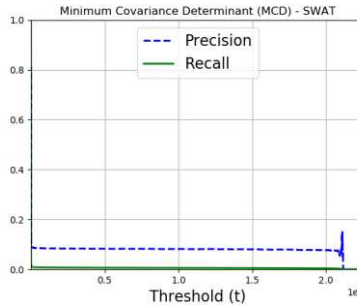
FRAUD

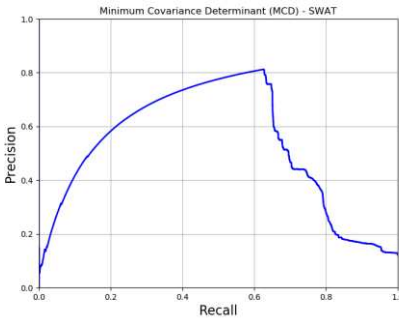


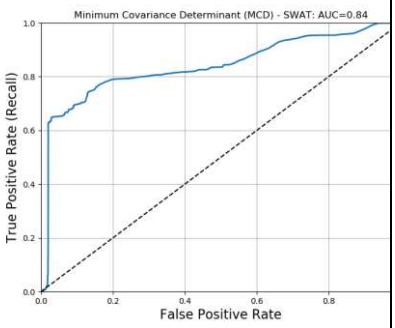




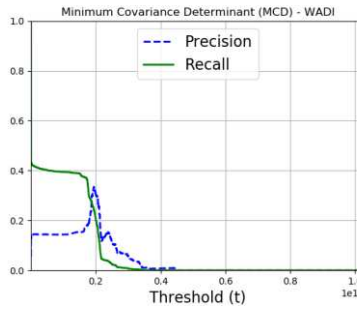
SECURE WATER TREATMENT TESTBED (SWAT)

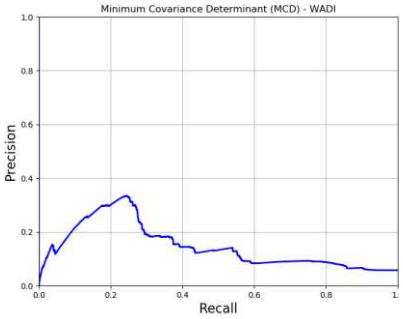


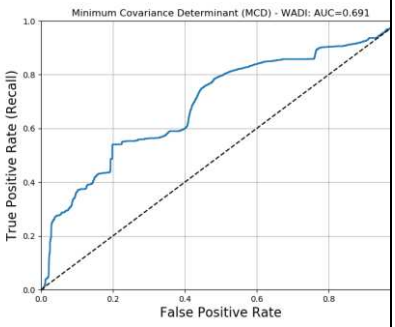




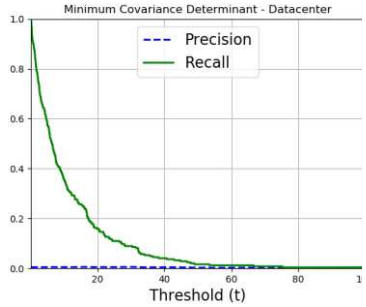
WATER DISTRIBUTION TESTBED (WADI)

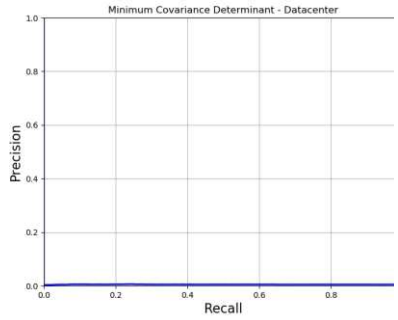






DATACENTER





d

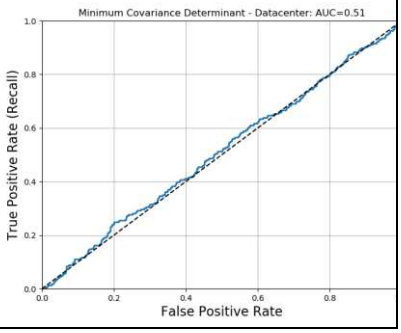
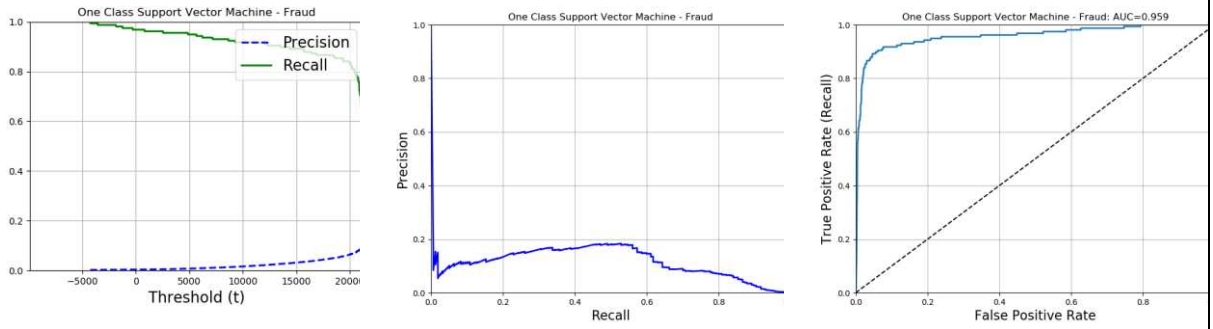
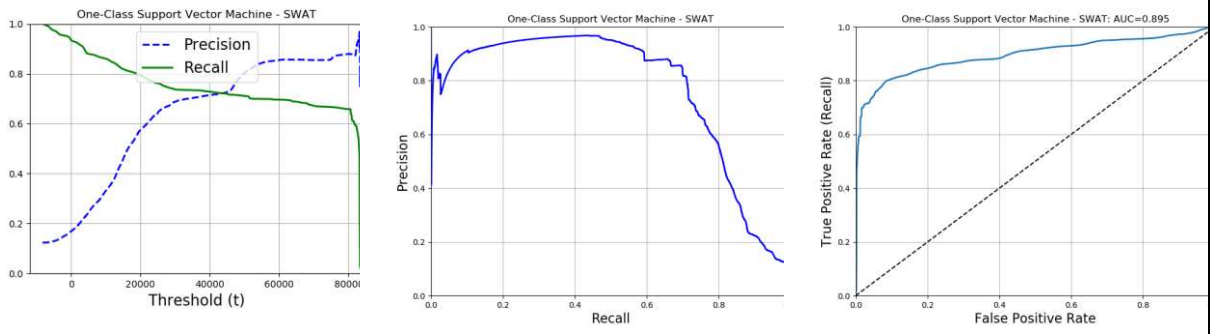


Figure 23: Minimum Covariance Determinant (MCD)

FRAUD



SECURE WATER TREATMENT TESTBED (SWAT)



WATER DISTRIBUTION TESTBED (WADI)

DNF – DID NOT FINISH

DATACENTER

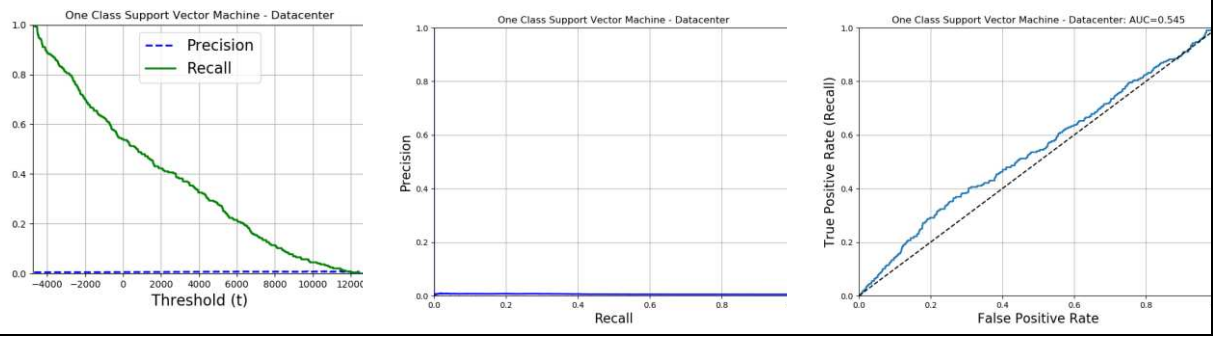
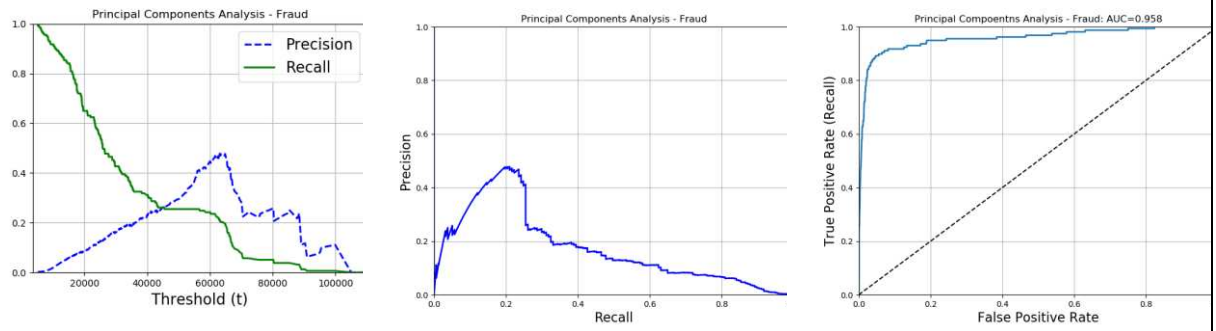
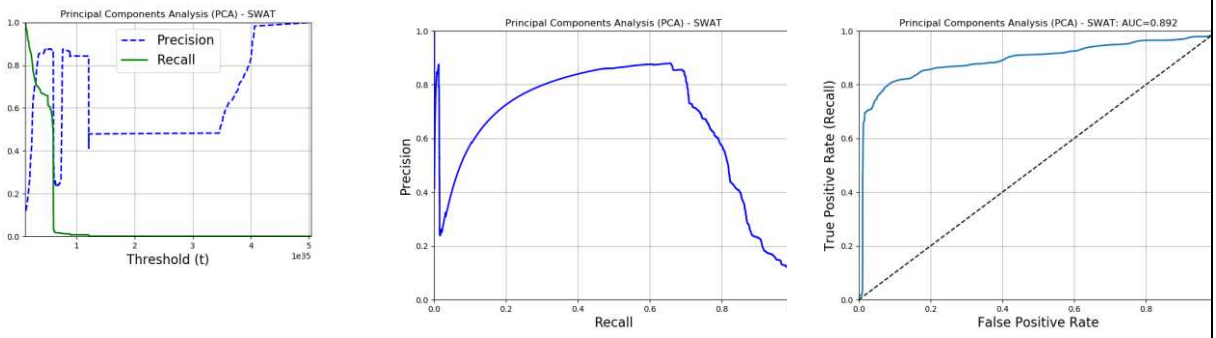


Figure 24: One-Class Support Vector Machines (OC-SVM)

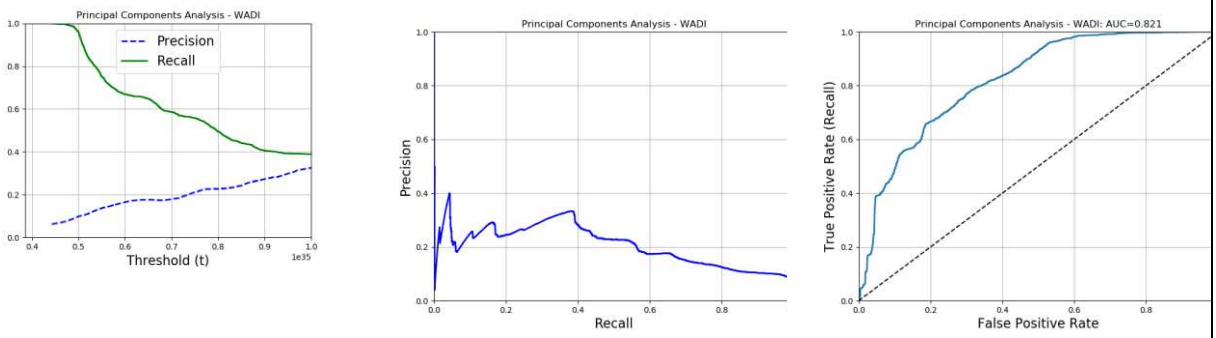
FRAUD



SECURE WATER TREATMENT TESTBED (SWAT)



WATER DISTRIBUTION TESTBED (WADI)



DATACENTER

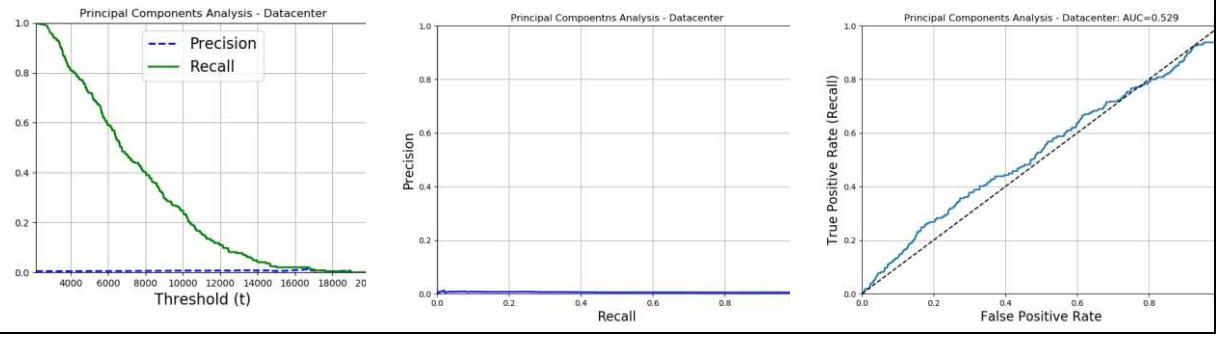


Figure 25: Principal Components Analysis (PCA)

CHAPTER 6 – DEEP NEURAL NETWORK (DNN) ANOMALY DETECTION

6.1 Background

DNNs experienced a resurgence in popularity beginning in the early 2000s that continues today. This resurgence, in part, has been a result of the availability of low-cost computer infrastructure, in part by the emergence of new algorithms, and in part, because of the generation of large datasets from IoT and SCADA devices required new analytic techniques.

Multiple network architectures and associated algorithms are included under the umbrella of DNNs. These architectures include feedforward neural networks (FFNs), recurrent neural networks (RNN), convolutional neural networks (CNN), and generative adversarial networks (GAN). DNN architectures and algorithms have been employed singularly or in combination to address complex applications such as image captioning, natural language processing, speech recognition, transfer learning, and sentiment analysis. For example, RNNs and CNNs have been successfully combined to produce an ensemble algorithm that can learn complex language embeddings for language translation [69]. Many of the mobile digital assistant applications are based on ensemble techniques targeted to smaller smartphone processors.

There has also been a resurgence in research in advanced techniques for anomaly detection. Studies have applied a variety of innovative algorithms to anomaly detection problems producing results superior to the TML techniques described in Chapters 4-5 [70]. These algorithms have borrowed heavily from the DNN neural machine translation and image processing literature. All of the types of DNN architectures described above have also been adapted to address anomaly detection problems.

In designing a DNN architecture, the choice of the number of hidden layers, the number of processing nodes within a hidden layer, the nonlinear activation function, the form of the

output, and the overall parameter estimation strategy are tantamount. Parameters are estimated with training data using an optimizer [71] that implements variants of the backpropagation algorithm [72] and various styles of stochastic gradient descent [73]. Practical estimation of DNNs also requires a set of choices upfront regarding the model hyperparameters. There are DNN meta-models designed to optimize the hyperparameter selection criteria [74]. Important hyperparameter decisions include the setup, initialization, and normalization of the architecture [75] and [76]; the selection of gradient descent optimization techniques such as the ADAM optimizer [71] and adjustments to the backpropagation algorithm (e.g., gradient clipping) to increase stability and prevent exploding or vanishing gradients; the approach to model overfitting including penalty-based regularization, early stopping, and dropout [77]; the decision regarding the treatment of dynamic learning-rates through learning rate decay, momentum [78], and other techniques; and the selection of GPU hardware acceleration and parallel computing infrastructure. However, the analysis of these hyperparameters choices on the formulation and performance of anomaly detection algorithms, albeit important, is beyond the scope of this research.

Below is a discussion of six (6) candidate DNN-based architectures that could support spatiotemporal anomaly detection stream processing. These six architectures are: a) Shallow/Deep Autoencoders (SDA), b) Variational Autoencoders (VAE), c) Deep Autoencoding Gaussian Mixture Models (DA-GMM), d) Generative Adversarial Networks (GAN), e) Encoding-Decoding Recurrent Neural Networks (ED-RNN), and f) Encoding-Decoding One-Dimensional Convolutional Neural Network (ED-1D-CNN). A few specialized DNN architectures have been used in anomaly detection studies. These other architectures were rejected from consideration either because the core technique lacked intrinsic support for

unsupervised learning, lacked the theoretical underpinnings that would justify use in an anomaly detection study, or could not be reasonably adapted to support time-dependent spatiotemporal streaming data.

Architectures #2 (VAE) and #4 (GAN) are members of a class of DNNs called generative models. With generative models, the network learns the model's probability distribution from the training data and generates new samples that can be used to complement the anomaly detection identification process. VAEs use approximate density estimation techniques, while GANs use implicit density estimation techniques [79]. Density estimation is a core problem of unsupervised learning and has been at the forefront of new approaches to anomaly detection.

There is an essential distinction between an architecture and an anomaly detection algorithm. The architecture describes the relationships between the various components of the system, the approach to parameters estimation, and the constraints placed on the system. The anomaly detection algorithm, however, goes deeper by providing a specific algorithm to designate the existence or nonexistence of an anomaly. For each of the six architectures described below, the associated anomaly detection algorithm is presented using pseudocode, a notation resembling a simplified programming language.

6.2 Architecture #1: Shallow/Deep Autoencoder (SDA)

SDAs are an unsupervised technique designed to encode data and reduce dimensionality efficiently. Traditionally, SDAs have been used for dataset cleansing and noise reduction similar to the capabilities provided by linear, statistically-based Principal Components Analysis (PCA) and Singular Value Decomposition (SVD). However, recent discoveries have suggested that SDAs can also provide novel anomaly detection capabilities.

The concepts of representational learning [80] and feature extraction are essential here. DNNs at their core are universal nonlinear function estimators and feature extractors. Similarly, SDAs can extract features from complex multivariate data through encodings and dimensionality reduction and can represent those features as vectors. These feature vectors, in turn, can be entered into other TML and DNN architectures designed for anomaly detection using ensemble algorithms and heuristic techniques.

SDAs strive to replicate a set of inputs through a sequential process known as encoding and decoding. The encoding process produces a minimalist representation by reducing the number of dimensions of the input data. This reduced dimensionality is then processed by the decoder to reproduce the original inputs. The idea is to preserve as much information as possible through the encoding and decoding process. An anomaly would be identified if the decoding cannot faithfully reproduce the original inputs to some predetermined threshold level.

Figure 26 depicts an example of a deep SDA with five (5) inputs, three (3) hidden layers with a compressed representation (middle hidden layer) of two nodes. These hidden nodes are also referred to as feature vectors because the SDA has extracted features from the raw data. The goal is to replicate or copy the vector of inputs x_ℓ^t. Each region ℓ has a separate instance of the model; there is no sharing of parameters across regions. The performance objective is to minimize the reconstruction error, the error between the original data and the reconstructed data. This reconstruction error is also considered the anomaly score.

The architecture of shallow and deep autoencoders are similar; the difference is that shallow autoencoders are designed to have only a single hidden layer while deep autoencoders are designed with two or more hidden layers. Otherwise, there is no practical difference between

a shallow and deep autoencoder. Also, note that autoencoding is an unsupervised technique because labeled data is not used in the training process.

The parameters of the encoder and decoder are estimated using a traditional FFNs with an optimizer that implements the backpropagation algorithm. However, an SDA does not have a clear concept of sequence or time embedded into the architecture; there is no time component shown in Figure 26. Therefore, a practical implementation of an SDA architecture will require a model adaptation to incorporate the temporal dimension. For example, the inputs into the SDA might include the data from t-1 (x_ℓ^{t-1}), from t-2 (x_ℓ^{t-2}), and from t-3 (x_ℓ^{t-3}). Unfortunately, in practice, this approach may be intractable in applications with long-term temporal relationships. With architecture #5, RNNs are a more natural approach to incorporating time-dependent data but with much higher compute resource requirements.

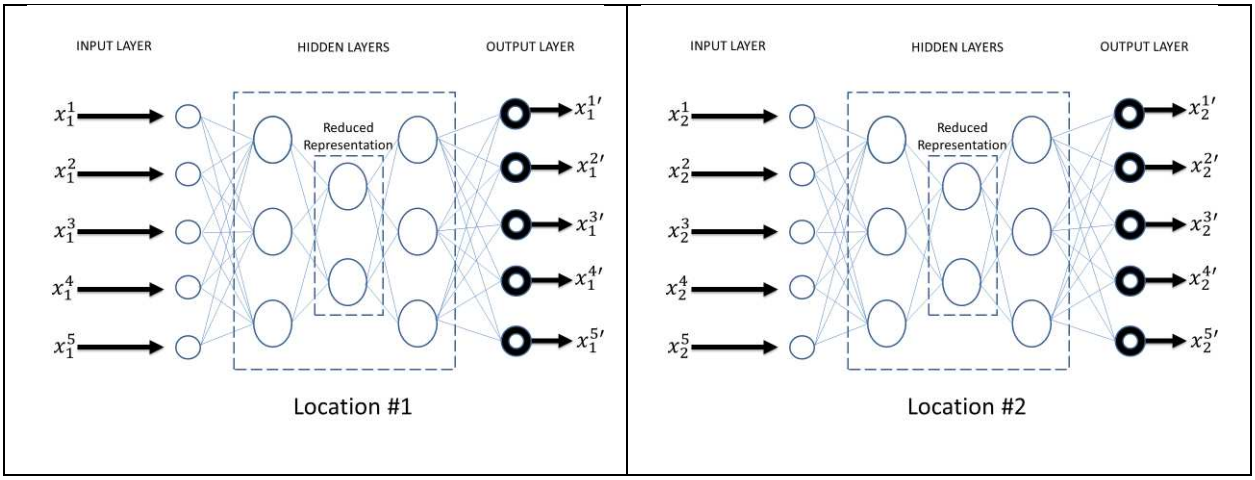


Figure 26: Shallow/Deep Autoencoder (SDA)

Mathematically, omitting the location subscript *l* for clarity, assume there are N multivariate samples from the training dataset ($x^1, x^2, \ldots, x^N$), with an encoder function f_θ, then for each sample *n* from the training dataset, the hidden feature vector is given by:

$$h^{\mathrm{n}} = f_\theta(x^n) \tag{6.1}$$

The decoder function, which maps or reconstructs the hidden feature vector back to the original inputs, is given by $g_{\emptyset}$:

$$r^n = g_{\emptyset}(h^n) \tag{6.2}$$

The set of parameters $\theta \; and \; \emptyset$ are learned by a DNN which attempts to minimize the reconstruction error over the entire set of training samples $(x^1, x^2, \ldots, x^N)$. The reconstruction error (RE) is the anomaly score. The reduction in dimension from an input size of five (5) to two (2) nodes forces the DNN to extract only the most salient features and learn the set of parameters $\theta \; and \; \emptyset$. This bottleneck produced by the encoder also forces the network to learn an efficient compression of the data into a lower-dimensional space. When data is encoded, only the regularities in the data are captured; irregularities and noise are ignored. So, the goal of the decoder is to minimize the RE or loss across all non-anomalous training samples given by equations (6.3):

$$Min\text{: } Reconstruction\ Error\ (\theta, \emptyset) = \sum_n RE\left(x^n, g_\theta(h^n)\right) \tag{6.3}$$

or equivalently

$$Min\text{: } Reconstruction\ Error\ (\theta, \emptyset) = \sum_n RE(x^n, g_\theta(f_\theta(x^n)))$$

Let s_f be the nonlinear activation function for the encoder, let s_g be the nonlinear activation function for the decoder, b is the encoder bias vector, d is the decoder bias vector, DE is the encoder weight matrix, and EN is the decoder weight matrix. The equations of the autoencoder for each training sample are given by equations (6.4):

$$\begin{aligned} f_\theta(x^n) &= s_f(DEx^n + b) \\ g_{\emptyset}(h^n) &= s_g(ENh^n + d) \end{aligned} \tag{6.4}$$

So, an autoencoder attempts to minimize the loss of information from the encoding process across all training samples, as seen in equation (6.5):

$$Min: Reconstruction\ Error\ (\emptyset) = \sum_{n=1}^{N} \left(x^n - g_\emptyset \left(s_f (DEx^n + b) \right) \right) \tag{6.5}$$

and substituting:

$$Min: Reconstruction\ Error\ (\theta, \emptyset) = \sum_{n=1}^{N} \left(x^n - s_g \left(EN(s_f (DEx^n + b)) + d \right) \right) \tag{6.6}$$

or in terms of Figure 26, the goal is to minimize the reconstruction error $\|x - x'\|$, which is also known as the 'L1' norm.

As an aside, a norm is a function that maps vectors to non-negative values and measures the size of a vector. The L^p norm is given by equation 6.7, where $p \in \mathbb{R}, p \geq 1$. [81]

$$\|x\|_p = (\sum_i |x_i|^p)^{1/p} \tag{6.7}$$

The L^2 norm, with p=2, is known as the Euclidean norm, which is the distance from the origin to the point defined by x. The squared L^2 norm, $x^T x$, is easier to work with mathematically and is often mentioned in DNN algorithms. Note that the higher the norm index, the greater the emphasis on high values and the less the emphasis on low values. Other distance measures used in DNN cost functions include the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE).

6.2.1 SDA Anomaly Detection Algorithm

Algorithm 1 in Figure 27 displays the pseudocode for the SDA anomaly detection algorithm. The algorithm is based on a reconstruction error noted in the equations above. The parameters are estimated using only the non-anomalous samples in the training set as the underlying premise is that all of the methods are unsupervised, supporting unlabeled data. Note that the number of anomalies is determined by the threshold α, which is application dependent;

the higher the threshold, the fewer the number of data points that are designated as anomalous since higher scores mean a higher probability of an anomaly.

Algorithm 1: Autoencoder Anomaly Detection Algorithm	
INPUT:	**Normal dataset X, Anomalous dataset x^i, i = 1,…,N, threshold α**
OUTPUT:	**reconstruction error $\|\|x - x'\|\|$**

STEPS:	$\emptyset, \theta$ ← **train an SDA using the normal dataset X**
	
	for i = 1 to N do
	reconstruction error(i) = $\|\|x^i - g_\emptyset\left(f_\theta(x^i)\right)\|\|$
	if reconstruction error(i) > α then
	x^i **is an anomaly**
	else
	x^i **is not an anomaly**
	endif
	endfor

Figure 27: SDA Anomaly Detection Algorithm

6.3 Architecture #2: Variational Autoencoder (VAE)

A VAE [82] is a generative model that outputs a probability estimate rather than a reconstruction error as the anomaly score [83]. The SDA produces a numeric vector in the hidden layer that represents the set of learned or extracted features of the data. A VAE also extracts and recreates the latent features of a problem domain, but uses a probabilistic approach. When decoding from these encoded features, sampling is performed from the encoded statistical distribution to create the decoded output. The advantage of a VAE over the SDA is that probabilities tend to be more interpretable than absolute reconstructive errors. However, both approaches still require an arbitrary threshold value α to binary classify new samples as anomalous or non-anomalous.

Figure 28 depicts a simplified VAE encoder and decoder network. Each input feature x is assumed to follow a Gaussian distribution. The encoder, $q_\theta(z|x)$ encodes inputs x and outputs to Z, a Gaussian multivariate vector of latent or hidden features. The decoder, $p_\emptyset(x|z)$, draws from this Gaussian distribution and regenerates the inputs x in x'. The key idea is to determine

the probability that x' was generated from Z. If this probability is low, and depending on the threshold level α, the sample is deemed to be an anomaly. Note that unlike the SDA, once training is completed and the parameters θ *and* $\emptyset$ are estimated, the decoder is not used in the anomaly detection algorithm.

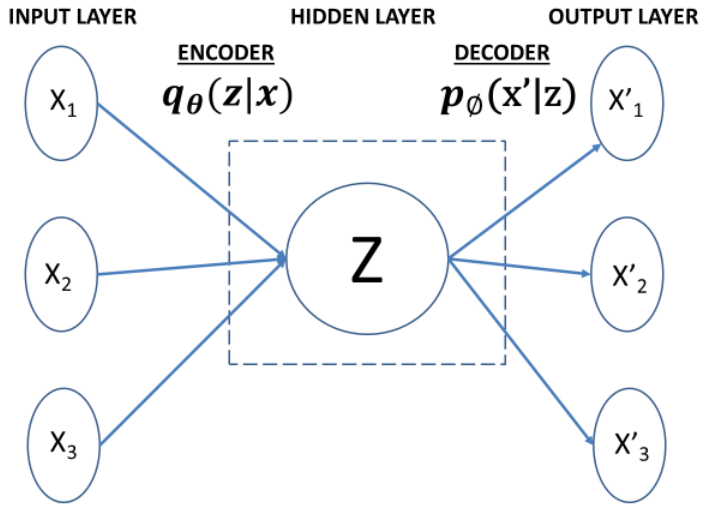


Figure 28: Variational Autoencoder (VAE)

Consider the encoder network $q_\theta(z|x)$. Over the set of multivariate Gaussian training samples, the output of the encoding DNN is a vector Z with mean $\mu_{z|x}$ and diagonal covariance of $\sum_{z|x}$. Similarly, the output of the decoder network $p_\emptyset(x|z)$is given by mean $\mu_{x|z}$and diagonal covariance of $\sum_{x|z}$. Both the encoder and decoder are probabilistic. If sample Z is generated from the training samples x, then $z|x \sim N(\mu_{z|x}, \sum_{z|x})$ and $x|z \sim N(\mu_{x|z}, \sum_{x|z})$. The goal is to estimate $p_\emptyset(x|z)$ to determine if the sample is an anomaly. To do so, estimate the (log) data likelihood of the i[th] training sample, as shown in equation 6.8 below, by taking the expected value with respect to z. Note that $p_\emptyset(x^i)$ does not depend on z.

$$\log p_\emptyset(x^i) = E_{z\sim q_\theta(z|x^i)}[\log p_\emptyset(x^i)] \tag{6.8}$$

Using Bayes theorem, we derive equation (6.9):

$$\log p_{\emptyset}(x^i) = E_{z\sim q_\theta}[\log \frac{p_{\emptyset}(x^i|z)p_{\emptyset}(z)}{p_{\emptyset}(z|x^i)}] \tag{6.9}$$

multiplying by a constant $\frac{q_\theta(z|x^i)}{q_\theta(z|x^i)}$, we get equation (6.10):

$$\log p_{\emptyset}(x^i) = E_{z\sim q_\theta}[\log \frac{p_{\emptyset}(x^i|z)p_{\emptyset}(z)}{p_{\emptyset}(z|x^i)} \frac{q_\theta(z|x^i)}{q_\theta(z|x^i)}] \tag{6.10}$$

and taking logarithms, we get (6.11):

$$\log p_{\emptyset}(x^i) = E_z[\log p_{\emptyset}(x^i|z)] - E_z\left[\log \frac{q_\theta(z|x^i)}{p_{\emptyset}(z)}\right] + E_z[log \frac{q_\theta(z|x^i)}{p_{\emptyset}(z|x^i)}] \tag{6.11}$$

which is equivalent to (6.12):

$$\log p_{\emptyset}(x^i) = E_z[\log p_{\emptyset}(x^i|z)] - D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z)) + D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z|x^i)) \tag{6.12}$$

where K-L is the Kullback-Leibler Divergence, a method to measure the difference between two probability distributions. K-L divergence is always >=0. Note that the term $E_z[\log p_{\emptyset}(x^i|z)]$ is the decoder network that can be estimated through sampling and $D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z))$ is the K-L term between the Gaussians for the encoder and the z prior. Unfortunately, the term $D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z|x^i))$ from equation (6.12) is intractable and cannot be estimated because every z cannot be computed given the finite number of samples x^i.

Since we want to maximize the data likelihood of $\log p_{\emptyset}(x^i)$, we can drop the intractable term $D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z|x^i))$ from (6.12) to get (6.13):

$$\mathcal{L}(x^i, \emptyset, \theta) = E_z[\log p_{\emptyset}(x^i|z)] - D_{KL}(q_\theta(z|x^i)||p_{\emptyset}(z)) \tag{6.13}$$

Equation (6.13) is tractable and becomes the lower bound of $\log p_{\emptyset}(x^i)$ since K-L >=0. Gradient descent can be used to optimize $\emptyset$ *and* θ to maximize the likelihood of the lower bound $\mathcal{L}(x^i, \emptyset, \theta)$. $\mathcal{L}(x^i, \emptyset, \theta)$ is known as the variational lower bound, or 'ELBO' for short. During

VAE training, the goal is to maximize 'ELBO,' the lower bound across the entire set of training examples.

6.3.1 An Aside on Kullback-Liebler (K-L) Divergence

The K-L divergence is a measure of the difference between two probability distributions. The basic idea behind K-L divergence is derived from three fundamental concepts in information theory: (1) unlikely events provide higher information content than unlikely events, (2) there is no information gained from the occurrence of a known event, and (3) the occurrence of independent events provides additive information content. The self-information I of event x is given by:

$$I(x) = -\log P(x) \tag{6.14}$$

Given two probability distributions P(x) and Q(x), the K-L divergence measure is given by $D_{KL}(\mathrm{P}\|\mathrm{Q})$:

$$D_{KL}(\mathrm{P}||\mathrm{Q}) = E_{x\sim P}[logP(x) - logQ(x)] \tag{6.15}$$

Note that K-L divergence is not a true distance measure since the metric is not symmetric: $D_{KL}(\mathrm{P}||\mathrm{Q}) \neq D_{LK}(\mathrm{P}||\mathrm{Q})$. Also, while the distance measures are generally thought of as a physical distance in 3D space, the distance measure is applied to multivariate data that does not represent a physical distance. Barz et al. [84] provide an unsupervised spatiotemporal anomaly detection algorithm called 'Maximally Divergent Intervals' (MDI), which is based on high K-L divergence compared to all other data.

6.3.2 VAE Anomaly Detection Algorithm

The VAE anomaly detection algorithm is adapted from [82] and is shown in Figure 29. The algorithm is similar to the SDA anomaly detection algorithms except that reconstruction probability is used instead of the reconstruction error.

Algorithm 2: Variational Autoencoder Anomaly Detection Algorithm	
INPUT:	Normal dataset X, Anomalous dataset x^i i = 1,…,N threshold α
OUTPUT:	Reconstruction probability $p_\emptyset(x\|x')$

STEPS:	$\emptyset, \theta$ ← train a VAE using the normal dataset X
	
	for i = 1 to N do
	$\mu_{z^i}\sigma_{z^i} = q_\theta(z\|x^i)$ # from the encoder
	Draw L samples from $z \sim N(\mu_{z^i}\sigma_{z^i})$
	for l = 1 to L
	$\mu_{z'^{(i}}\sigma_{z^i} = p_\emptyset(x^i\|z^{(i,l)})$
	end for
	reconstruction probability(i)= $\frac{1}{L}\sum_1^L p_\emptyset(x^i\|\mu_{x'^{(i,l)}}\sigma_{x'^{(i,l)}})$
	if reconstruction probability(i) < α then
	x^i is an anomaly
	else
	x^i is not an anomaly
	endif
	endfor

Figure 29: VAE Anomaly Detection Algorithm

6.4 Architecture #3: Deep Autoencoding Gaussian Mixture Model (DA-GMM)

This discussion of the deep autoencoding Gaussian mixture models for unsupervised anomaly detection is based on [85]. DA-GMM is an unsupervised, anomaly detection approach that is an extension of the deep autoencoding model. As background, a mixture model is a probabilistic model designed to represent the existence of subpopulations within an overall population. For example, anomalous points within a normal population can be modeled as a normal distribution subpopulation of anomalous points. A GMM learns and assigns points to these subpopulations automatically. A GMM is unsupervised because these subpopulations are unknown and are assigned by the underlying model.

The DA-GMM shown in Figure 30 is composed of two connected networks, a compression network, and an estimation network. The compression network is similar to the SDA network and conducts dimensionality reduction and feature extraction. The compression network feeds the estimation network, which predicts the probability $\hat{\pi}$ that the reduced representation indeed represents the true data. The DA-GMM algorithm jointly optimizes the

parameters of the autoencoder and the mixture model simultaneously rather than sequentially. Through this joint optimization procedure, anomaly detection accuracy is improved by 10-15%, according to [85].

Note the compression network contains two sources of features: (1) the reduced, low-dimensional representations learned by the autoencoder, and (2) the features derived from the reconstruction error. The compression (autoencoding) network feeds the estimation network that takes the reduced dimensionality of the inputs and outputs mixture membership prediction (known as the likelihood/energy) for each sample. Gaussian mixture density estimation procedures are beyond the scope of this paper but are discussed in detail in Section 3.3 of [85]. So, to summarize, a DA-GMM model is essentially the combination of an SDA (architecture #1) with an integrated Gaussian mixture back-end model.

The DA-GMM objective function is given by (6.16):

$$J(\theta_c, \theta_d, \theta_m) = \frac{1}{N}\sum_{i=1}^{N} L(x_i x_i') + \frac{\lambda_1}{N}\sum_{i=1}^{N} E(z_i) + \lambda_2 P(\widehat{\Sigma}) \tag{6.16}$$

where $L(x_i x_i')$ is the loss function (e.g., L_2 norm) from the reconstruction error from the autoencoder in the compression network; $E(z_i)$ is the model of the probabilities observed from the input data, and λ_1 and λ_{12} are meta-parameters in DA-GMM.

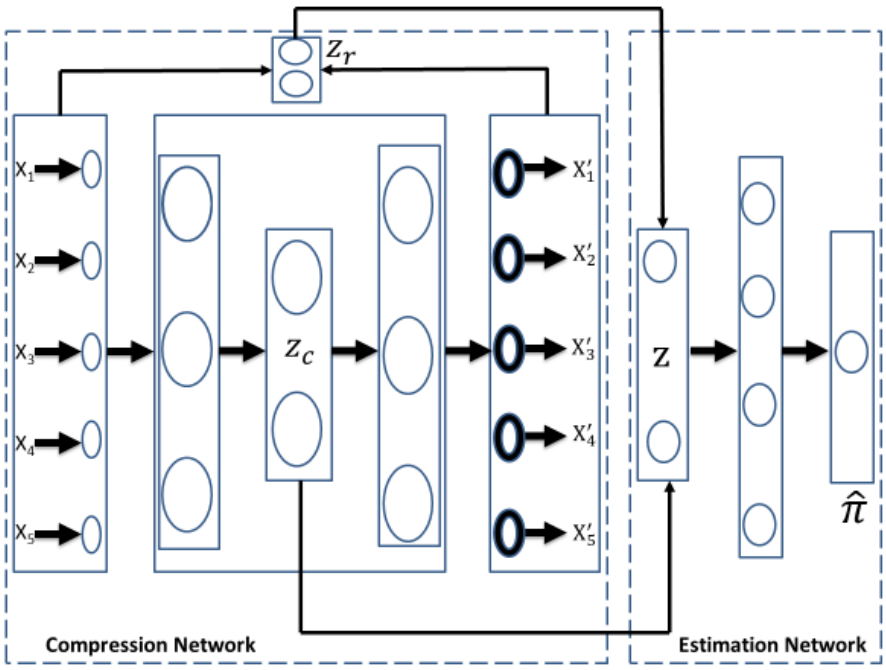


Figure 30: Deep Autoencoding Gaussian Mixture Model (DA-GMM)

6.4.1 DA-GMM Anomaly Detection Algorithm

Algorithm 3 in Figure 31 provides the DA-GMM anomaly detection algorithm. The estimation network utilizes a multi-layer DNN to predict the mixture membership of each sample and calculates the likelihood/energy values. Samples with high energy above the designated threshold are deemed anomalies. Because the DA-GMM supports a classification problem, the standard evaluation metrics such as Precision, Recall, and F_1 are applicable. The DA-GMM paper designates the highest 20 percent in terms of 'energy' of all samples is marked as anomalies. Note that by varying the energy threshold value, a standard ROC curve could be generated and an AUC metric calculated. Because DA-GMM is probabilistic, each run will produce different results, sometimes dramatically different. Results are averaged across twenty (20) executions.

Algorithm 3: Deep Autoencoder Gaussian Mixture Model Anomaly Detection Algorithm	
INPUT:	Normal dataset X, Anomalous dataset x^i i = 1,…,N threshold α, K=number of mixture components
OUTPUT:	Sample likelihood/energy

STEPS:	$\theta_c, \theta_d, \theta_m$ ← train a DA-GMM using the normal dataset X
	
	Estimate the parameters in GMM, including:
	$\emptyset$: *Mixture Component Distribution*
	μ: Mixture Means
	$\sum$: Mixture Covariance
	z: Low Dimensional Representations
	
	for i = 1 to N do
	calculate z_i
	sample energy(i,z_i)= $-log(\sum_{k=1}^{K} \emptyset_k \frac{exp(-\frac{1}{2}(z-\mu_k)T \sum_k^{-1}(z-\mu_k)}{\sqrt{\lvert 2\pi\sum_k \rvert}})$
	if sample energy(i, z_i) > α then
	x^i is an anomaly
	else
	x^i is not an anomaly
	endif
	endfor

Figure 31: DA-GMM Anomaly Detection Algorithm

6.5 Architecture #4: Generative Adversarial Network (GAN)

A GAN [86] is a generative model designed initially for image generation and is represented in Figure 32. Unlike VAEs and DA-GMMs, GANs are not based on probability density models but a two-player game-theoretic approach. Each GAN consists of two competing DNNs, a generator G with parameters θ_g and a discriminator D with parameters θ_d, which are trained simultaneously and which learns to distinguish between real and fake or anomalous data. G tries to fake the discriminator by generating realistic data, while D tries to distinguish between real and fake data. After the model is trained, anomalies are identified by D as fake data. As with all classification problems, the anomaly determination is dependent on the threshold value of D. Higher threshold values will result in fewer true positives but also fewer false positive anomaly designation. There is always a trade-off between true and false positives given cutoff value.

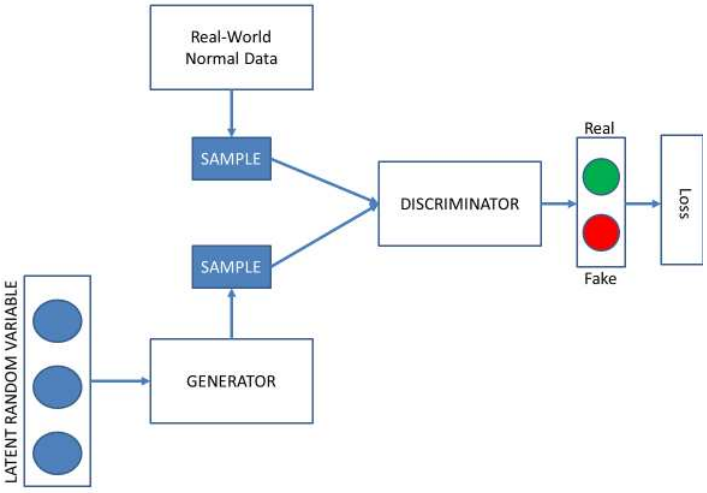


Figure 32: Generative Adversarial Network (GAN)

Consider equation (6.17), which is a formulation of a minimax game, also known as a zero-sum game:

$$\min_{\theta_g} \max_{\theta_d} [\, E_{x\sim p(data)} \log D_{\theta_{d,}}(x) \; + \; E_{x\sim p} \log \left(1 - D_{\theta_{d,}}\left(G_{\theta_g}(z)\right)\right)] \tag{6.17}$$

Note that $D_{\theta_{d,}}(x)$ is the discriminator output for real data while $D_{\theta_{d,}}\left(G_{\theta_g}(z)\right)$ is the discriminator output for the generated fake data $G_{\theta_g}(z)$. The discriminator D outputs a likelihood in the range (0,1), where one (1) is real, and zero (0) is fake data. D strives to maximize the objective such that $D_{\theta_{d,}}(x)$ is close to one and $D_{\theta_{d,}}\left(G_{\theta_g}(z)\right)$ is close to zero, while G strives to minimize the objective such that $D_{\theta_{d,}}\left(G_{\theta_g}(z)\right)$ is close to 1. GAN training includes an optimizer with minibatch gradient descent, with the backpropagation algorithm, is alternately applied between the discriminator shown in equation (6.18) and the generator shown in equation (6.19). Equilibrium is reached when either the generator or the discriminator will not alter their parameters regardless of what the other does, which is also known as a Nash equilibrium.

$$\max_{\theta_d}[\, E_{x\sim p(data)} log D_{\theta_d,}(x) + \, E_{x\sim p(z)} \log\,(1 - D_{\theta_d,}\Big(G_{\theta_g}(z)\Big))] \tag{6.18}$$

$$\min_{\theta_g} E_{x\sim p(z)} \log\,(1 - D_{\theta_d,}\Big(G_{\theta_g}(z)\Big)) \tag{6.19}$$

GANs are challenging to train, in part because two DNNs are requiring two (large) sets of choices regarding hyperparameters and in part because of the inability to learn the model parameters. Learning may be difficult because of the mode collapse problem, where there is a tendency by the generator to explore only a limited subset of plausible solutions, or because the discriminator may overpower the generator, or visa-versa causing overfitting. GANs also exhibit bouts of non-convergence where the model parameters oscillate and destabilize.

6.5.1 GAN Anomaly Detection Algorithm

Algorithm 4 in Figure 33 displays the anomaly detection algorithm associated with a GAN. The same process as network training is followed except that the model parameters are not re-estimated. The input data is passed through the generator network, followed by the discriminator network. An anomaly score (0,1) is calculated and compared against the threshold value. The threshold value is set where the probability of a fake is greater than .5

Algorithm 4: Generative Adversarial Network Anomaly Detection Algorithm	
INPUT:	**Normal dataset X, Anomalous dataset x^i i = 1,…,N threshold α**
OUTPUT:	**GAN Output Score (0,1)**

STEPS:	**for i = 1 to N do**
	Generate z^i from x^i
	Enter Generator Trained Network. Calculate $D_{\theta_d,}\Big(G_{\theta_g}(z)\Big)$
	output score(i) = $log D_{\theta_d,}(x)$
	if output score(i) < α then
	x^i **is an anomaly**
	else
	x^i **is not an anomaly**
	endif
	endfor

Figure 33: GAN Anomaly Detection Algorithm

6.6 Architecture #5: Encoding-Decoding Recurrent Neural Network (ED-RNN)

RNNs have been used extensively in anomaly detection applications and can be adapted to form an autoencoding-like network. In this architecture, the output of an RNN attempts to replicate the inputs similar to an SDA. An ED-RNN combines the best of both architectures, adding a temporal component to an otherwise static SDA. ED-RNN style architectures are heavily used in machine and speech translation applications today, although the temporal component is not time but rather a sequence of words or speech.

Figure 34 illustrates the basic variant of an ED-RNN. An RNN is a dynamic network that contains delays and operates on an ordered sequence of inputs. An RNN is represented as a directed graph along a temporal sequence. The self-loop or folded depiction is shown on the left, while the unfolded depiction is shown on the right. Both the folded and unfolded versions are conceptually identical. The subscripts represent location while the superscripts represent time or sequence. So, for example, x_1^{t-1} represents a multivariate input vector at location 1 at time t-1. Note that the hidden state $h_l^t$is a function of not only the current input vector x_l^t, but also the previous period's hidden state h_l^{t-1}. This model structure provides the long-term memory of the network. Long-term memory is a desirable characteristic in anomaly detection architectures for uncovering complex temporal relationships.

An ED-RNN is sensitive to the order of the data, supports long-term temporal patterns, and the identification of global anomalies. An ED-RNN is not appropriate for the identification of local anomalies. Like all DNN architectures, an ED-RNN can be combined with other architectures to form a hybrid architecture that could potentially identify both local and global anomalies.

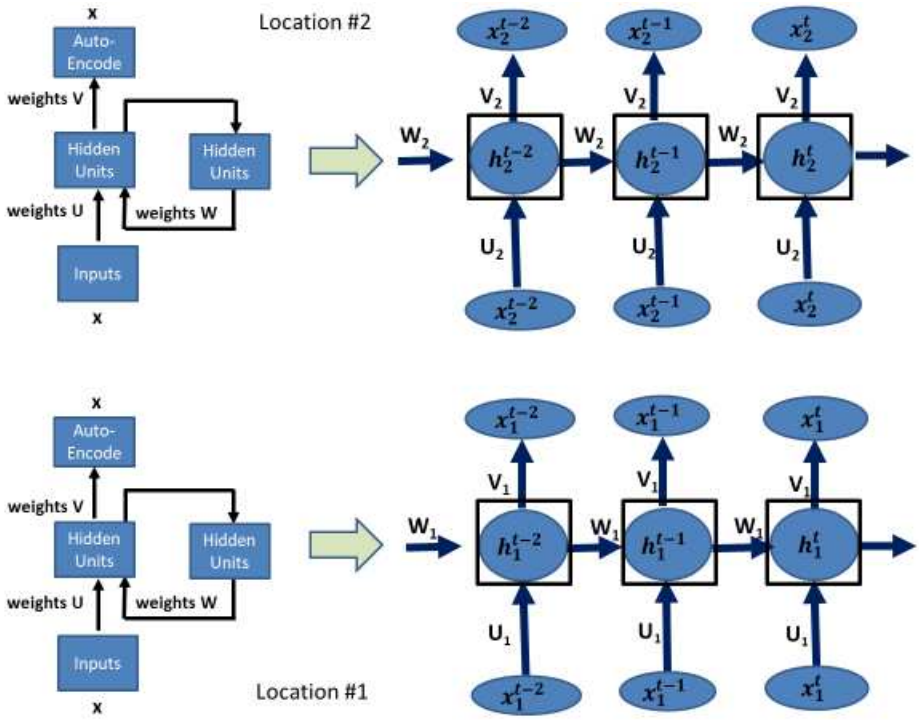


Figure 34: Encoding-Decoding Recurrent Neural Network (ED-RNN)

The goal of the autoencoder component is to replicate these three inputs by learning the hidden representation h, which includes features from not only the current inputs but also from the sequential history of inputs as captured by the RNN hidden units. Formally, the network equations for the RNN are given by equation (6.20), omitting the location subscripts for brevity:

$$h^t = f(h^{t-1}, x^t) \text{ or} \tag{6.20}$$

$$h^t = f(Wh^{t-1} + Ux^t)$$

where the hidden state is a nonlinear activation function $f(.,.)$. Popular activation functions $f(.,.)$ include rectified linear units, sigmoid, and tanh. The final output is another function $g()$ of the hidden states:

$$y^t = g(h^t), \text{ or equivalently} \tag{6.21}$$

$$y^t = g(f(Wh^{t-1} + Ux^t))$$

Note that the RNN weight matrices U, W, and V are shared across time-steps. Sharing means that the model weights are constrained by the optimization algorithm to be identical across time or sequence. Each location or geographic region, however, would have a separate instantiation of this architecture. Parameter sharing is desirable in RNNs to avoid many stability issues when estimating the model parameters with long temporal sequences. More importantly, the sharing of weights across time steps is theoretically grounded; there is no reason to believe that the underlying structural model drifts or is different across time within a given region. If the weight parameters were not shared across time, then each period would be a separate model. Note that the weight matrices U, W, V are not shared across locations in this model, meaning that each location is operating autonomously and subject to a different underlying model. Whether RNN weight matrices should be shared across locations is an application design question and also an open research issue in the DNN literature.

There are many variations in the EN-RNN architecture, including alternate connections [87], additional weight matrices, and different parameter constraints across time. RNN formulations handle variable-length input streams that are critical with time series or sequence data. Specialized versions of the backpropagation algorithm known as the backpropagation through time (BPTT) algorithm are required to estimate the parameters of the RNN. Unfortunately, the BPTT algorithm is processor-intensive, has difficulty learning long sequences (over a few hundred steps at best) due to the ubiquitous vanishing or exploding gradient problem [88] and may fail to converge. As a result of these estimation issues, architectural add-ons to RNNs such as Long Short-Term Memory (LSTM) [89] and Gated-Recurrent Units (GRU) have been developed and are almost always used. Even with these architectural add-ons, RNNs are long-running and processor-intensive [90]. For these reasons, the use of RNNs in streaming

anomaly detection applications is problematic. Note that the optimizer with the backpropagation algorithm used in the encoder-decoder component is less complicated, more stable, and significantly faster than the BPTT algorithm required by RNNs. Therefore, an SDA architecture is more performance compatible with streaming than an ED-RNN. However, conceptually, an ED-RNN is designed for multi-step temporal or sequential data, while SDAs are designed for snapshots or cross-sections.

Because of the many estimation and complexity issues associated with RNNs with LSTM and the difficulty of training these models [91], there has been a recent trend in the neural machine translation literature away from recurrent networks and towards attention-based networks, which is similar to the SDA architecture. The belief is that in most applications, and particular machine translation applications, long-term memory over many steps is not necessary and that the focus attention of network connections should be on recent or nearby memory. See Vaswani et al. [92] for more details. For a discussion of ED-RNN in the application of machine language translation, see [93].

6.6.1 ED-RNN Anomaly Detection Algorithm

Figure 35 displays the pseudocode for the anomaly detection algorithm associated with an ED-RNN. The algorithm is based on a multi-step reconstructive error. Using an RNN, a multistep reconstructive error is made by comparing actual with predicted values. If the error between the actual/predicted reconstructive exceeds a threshold α, then the multi-step collectively is deemed an anomaly. The approach to multi-step reconstructive step thresholds is application-defined. The threshold could be an average reconstructive error per-period, the sum over the period, or a mini-max value where the reconstructive error in any period cannot exceed a particular maximum threshold value.

Algorithm 5: Encoding-Decoding Recurrent Neural Network Anomaly Detection Algorithm	
INPUT:	Normal dataset X, Anomalous dataset x_i^t i = 1,…,N threshold α,time t at any given location (not shown) Reconstruction Sequence Window length: s Threshold Technique: minimum, maximum reconstruction error
OUTPUT:	Sum Reconstruction Error across Sequence Length

STEPS:	U, W, V ← train an EN-RNN using the normal dataset X
	
	for i = 1 to N do
	  for j = 0 to (s-1) do
	  reconstruction error(i) = $\|\|x_i^{t-j} - g(f(Wh^{t-j-1} + Ux^{t-j}))\|\|$
	  if reconstruction error(i) > α then
	  x_i^{t-j} is an anomaly
	  else
	  x_i^{t-j} is not an anomaly
	  endif
	  endfor
	endfor

Figure 35: ED-RNN Anomaly Detection Algorithm

6.7 Architecture #6: Encoding-Decoding 1D Convolutional Neural Network (ED-1D-CNN)

An ED-1D-CNN is similar to the ED-RNN architecture except that the RNN component is replaced by a one-dimensional convolutional neural network (1D-CNN). 2D and 3D CNNs are designed to operate convolutionally, extract features for local inputs, and are suited for computer vision problems. These same concepts can be applied to anomaly detection, except that the domain is 1D and is modified for sequence processing. An RNN captures long-term temporal patterns and supports the identification of global anomalies. With a 1D-CNN, local 1D patches, or subsequences, are extracted from the complete sequence. Unlike an RNN, these subsequences are location invariant. For this reason, a 1D-CNN captures local, translation invariant patterns and best supports the identification of local anomalies. When combined with an encoder and decoder, an anomaly detection algorithm can be designed. See Chollet [94] for an overview of the 1D-CNN architecture.

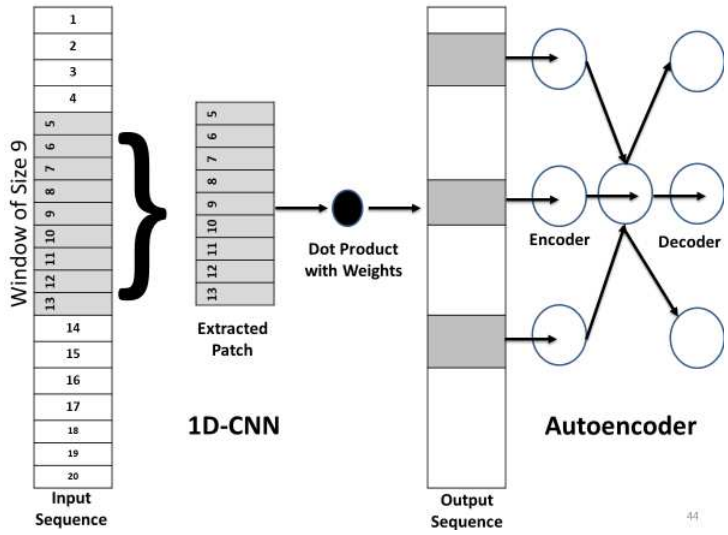


Figure 36: Encoding-Decoding One-Dimensional Recurrent Neural Network (ED-1D-CNN)

Figure 36 displays the architecture of an ED-1D-CNN. Rather than processing the entire sequence, ED-1D-CNN processes a convolutional window into the input sequence, extracts patches from that sequence (e.g., nine (9) steps), and learns a set of weights that minimizes the reconstruction error when feeding through the autoencoder. This results in more stable, lighter weight and faster DNN over the ED-RNN architecture, albeit at a potential loss of information from the convolution. Note that the ED-1D-CNN will capture sequence invariant local anomalies while the ED-RNN will capture global anomalies across the entire sequence.

6.7.1 ED-1D-CNN Anomaly Detection Algorithm

Figure 37 displays the ED-1D-CNN anomaly detection algorithm. This algorithm is similar in structure to the ED-RNN algorithm except that instead of an RNN preprocessing the inputs to the autoencoder, a 1D-CNN provides the inputs.

Algorithm 6: Encoding-Decoding One Dimensional Convolutional Neural Network Anomaly Detection Algorithm	
INPUT:	Normal dataset X, Anomalous dataset x_i^t i = 1,…,N threshold α,time t at any given location (not shown, convolutional window size w Reconstruction Sequence Window length: s Threshold Technique: minimum, maximum reconstruction error
OUTPUT:	Sum Reconstruction Error across Sequence Length

STEPS:	$W \leftarrow$ train an 1D-CNN using the normal dataset X
	
	for i = 1 to N do
	  for j = 0 to (s-1) do
	  reconstruction error(i) = $\|\|x_i^{t-j} - g(f(Wh^{t-j-1} + Ux^{t-j}))\|\|$
	  if reconstruction error(i) > α then
	  x_i^{t-j} is an anomaly
	  else
	  x_i^{t-j} is not an anomaly
	  endif
	  endfor
	endfor

Figure 37: ED-1D-CNN Anomaly Detection Algorithm

6.8 Architecture Summary

Table 15 provides a summary of the various architectures discussed in this chapter. The techniques range from very mature (RNN) to recent (DA-GMM). Only the DA-GMM was explicitly designed for unsupervised anomaly detection. However, both the SDA and VAE architectures were designed for unsupervised learning so that these techniques can easily be adapted. Adapting RNNs to unsupervised learning is more challenging since these techniques were designed for prediction and forecasting, a supervised problem. However, RNNs and, to some extent, SDAs are well supported in the open-source software community and have readily available software implementations. VAE also has support but primarily for image-related tasks. DA-GMM architectures, however, require a custom software implementation as these architectures are new and are not implemented in open-source software packages.

Suitability to adaptation for streaming architectures as well as performance is important considerations when evaluating alternative anomaly detection architectures. These architectures were designed to be trained offline, separate from deployment. Once the model is trained, the

parameters are deployed operationally and perhaps periodically re-estimated. While in many application domains, this workflow may be appropriate, for dynamic, high-tempo anomaly detection domains in conditions of concept drift, this approach is unsatisfactory. Concept drift occurs when the statistical properties of the target variable change over time in unforeseen ways. In these situations, online streaming applications need to be constantly adapting and learning.

Table 15: Comparison of Anomaly Detection Architectures

	SDA	VAE	DA-GMM	GAN	ED-RNN	ED-1D-CNN
Architecture #	1	2	3	4	5	6
Technology Core	Non-Parametric	Probabilistic	Gaussian Mixtures	Game Theory	Dynamic Systems	Image Processing
Built-In Temporal Support	No	No	No	No	Yes	Yes
Complexity	Low	Moderate	High	Moderate	High	Moderate
Estimation Technique	FeedForward Network	FeedForward Network	FeedForward Network	FeedForward + Convolutional	Recurrent Network	Convolutional Network
Anomaly Algorithm	Reconstruction Error	K-L Divergence	Probability Error	(0,1)	Reconstruction Error	Reconstruction
Largest Domain Usage	Image Processing	Image Processing	Anomaly Detection	Image Augmentation	Neural Machine Translation	Sequence Processing
Maturity Level	High	Moderate	Low	Moderate	High	Low
Reference	[55]	[83]	[85]	[86]	[93]	[94]

The six (6) architectures will undergo experimentation in Chapter 7 using the same datasets described in Chapter 3. This experimentation will inform the decision made and concerning the STADE architecture. The STADE architecture is designed to support streaming, near-real-time anomaly detection applications that can identify the concept drift described above. The STADE architecture is also designed to support pluggable algorithms. Pluggable means that algorithms can be interchanged for one another, or new algorithms inserted or tailored to specific domain requirements. STADE is designed to run multiple algorithms concurrently and combining the results using an ensemble approach that often produces better predictive performance compared to a single model.

6.9 Related Work

The authoritative sources for representational learning academic research are two major machine learning conferences, the International Conference on Machine Learning (ICML) and the Conference on Neural Information Processing Systems (NIPS). Virtually all machine and DNN research, including research-in-progress, are published at https://arxiv.org. The open-source community is rapidly expanding software support for various neural network architectures, led by Google™ with Tensorflow™ and Facebook™ with PyTorch™. The Open Neural Network Exchange (ONNX) is a vendor-independent industry consortium that is promoting a standard format to exchange neural network models and specifications. There has also been a measurable increase in the number of anomaly detection surveys and books published including Agrawal and Agrawal [40], Ariyaluran, et al. [95], Ahmed, Mahmood and Hu [96], Bhuyan, Bhattacharyya, and Kalita [97], and Bengio [98]. Goodfellow, Bengio, and Courville [81] is the definitive book on representational learning.

Two training datasets continue to dominate the representational learning-based anomaly detection literature. These datasets are the NMIST handwritten digital dataset for image processing, and the DARPA Knowledge Discovery and Data Mining Tools Competition 1999 dataset on network intrusion. Neither dataset provides true support for anomaly detection; the lack of high quality, readily accessible multivariate datasets has hampered research. Network intrusion detection is a heavily studied component of the much larger field of cybersecurity, and the techniques designed for identifying outliers in computer traffic may not be transferable to other anomaly detection domains. The two anomaly datasets used here, SWAT and WADI, are relatively new and have not appeared extensively in the literature.

6.9.1 Architecture #1: SDA Related Work

Architecture #1, shallow and deep autoencoders, is perhaps the most cited anomaly detection representational learning architecture today. Hawkins et al. [99], in a 2002 study, used a three hidden-layer replicator neural network to select anomalies in network intrusion and cancer datasets. Cordero et al. [100] used a similar replicator for intrusion detection of large internet traffic. The replicator terminology survived through 2016 when the term was replaced by autoencoder. Schreyer et al. [101] applies deep autoencoder networks to accounting records and show higher F_1 scores and lower false positives compared to current baseline methods in accounting. Chen et al. [102] use an ensemble of autoencoders (single architecture with different parameters) combined with a random edge and adaptive data sampling technique to improve performance. Socher et al. [103] use autoencoders for sentiment prediction, a topic that is further explored in the case study found in Chapter 11. Baldi [104] provides a mathematical basis for utilizing autoencoders for unsupervised representational learning models. Other autoencoder-based anomaly studies include [105], [106], [107], [108], [109], [110], [111], [112] and [113].

As noted in Section 4.2, SDAs do not have explicit support for temporal data; several DNN-based time-series studies have resorted to utilizing traditional feedforward nets or plain RNNs. See [114], [115], [116] and [117]. Chalapathy et al. [118] propose a one-class DNN model similar to the one-class SVM (OC-SVM) described in Chapter 4. Other interested temporal anomaly detection studies include an extreme event (i.e., anomaly) forecasting model for the Uber™ ride-sharing company using an RNN with LSTM [119] and a convolutional encoder-decoder model from Zhang et al. [120].

6.9.2 Architecture #2: VAE Related Work

Variational Autoencoders (VAE) is now the preferred approach to anomaly detection due to the support for the interpretability of the probabilistic anomaly scores. Interestingly, VAEs made popular for image processing are now losing popularity and being replaced by GANs. The initial formulation of VAEs by Kingma and Welling [83], a comprehensive tutorial on VAEs by Doersch [121], and a comparison of VAEs with SDAs with an evaluation on MNIST by An and Cho [82] have all contributed to the popularity of this type of autoencoder. VAEs are a type of generative model where the network learns the underlying data distribution in order to produce new data points with variations. Anomaly detection studies that have used VAEs include Solch et al. [122] and Xu et al. [123]. Studies applying VAEs to specific domains include intrusion detection in IoT [124], skin disease [125], and brain magnetic resonance imaging (MRI) images [126]. Kim et al. [127] combines a VAE with a CNN to predict anomalies in sensor time-series and finds improvement in performance. Borghesi et al. [128] apply VAEs to High-Performance Computing (HPC) anomalies, Walker et al. [129] to static images, Luo and Nagarajan [130] to wireless sensor networks anomalies, Oh and Yun [131] to sound data anomalies, Chalapathy et al. [132] to image data, and Guo et al. [133] on laboratory-generated sensor data.

6.9.3 Architecture #3: DA-GMM Related Work

The DA-DAGG model by Zong, et al. [85] described in detail in section 6.4.1 above combines the output of an autoencoder with a Gaussian Mixture Model under a two-step ensemble approach with improved results over the autoencoder results alone. This approach is new and has only recently appeared in the literature.

6.9.4 Architecture #4: Generative Adversarial Network (GAN)

GANs were initially designed to generate images for data augmentation but has recently been applied to non-image anomaly detection problems. There are many variants of a GAN, as described in [86]. The Wasserstein GAN is a method to evaluate the distance between two high-dimensional datasets known as the 'earth mover' (EM) distance and is used in [134]. This GAN inspects the distribution of each variable, real and fake. The GAN determines how much effort (in terms of mass times distance) is required to push the generated distribution into the shape of the real distribution. The Wasserstein GAN distance measure is the anomaly score. Multi-Discriminator GAN is another approach that includes a dense network for determining whether the generated samples are of sufficient quality (i.e., valid) and an autoencoder that serves as an anomaly detector. Other generative models have been combined with CNNs to address anomaly detection of sequence data [38] using the WADI and SWAT datasets. Buitrago et al. [135] demonstrate the use of GANs and VAEs to identify anomalies in the presence of unbalanced (low frequency) anomaly data. Sequence data is also supported by GANS, as discussed in [136]. Other models using GANS or related technologies include [137], [138], [139], [140], [141], [142], [143] and [144]. A highly readable overview of adversarial autoencoders is presented in [145]. Wang et al. [146] apply maximum likelihood estimation to GANS for anomaly detection.

6.9.5 Architecture #5: Encoding-Decoding Recurrent Neural Network (ED-RNN)

RNNs, LSTMs, and sequence-to-sequence models have been used heavily in the representational learning-based anomaly detection literature. Early research by Elman [147] provided the basis for RNN-based anomaly detection studies that followed. Sequence-to-sequence recurrent models described by Graves [148] are used in many time-series, image recognition, and other studies, including [149], [150]. Example recurrent neural network

applications include electrical load [151], computer system logs [152] and [153], video forgeries [154], network traffic [155] and [156], electrocardiography (ECG) signals [157], sensor networks [158] and time-series [159] anomaly detection. Bontemps et al. [160] and Thi, et al. [161] both use the KDD-Cup dataset to address collective anomaly detection where anomalies are identified by prediction errors from multiple consecutive time steps. Other ED-RNN models include [162], [163], [164], [165], [166] and [167]. A special type of neural network model known as a restricted Boltzmann machine was a popular technique a few years ago in several anomaly detection studies, including [168] [169], but is seldom used today. For a study that uses an RNN to detect anomalies in the SWAT database, see [170].

6.9.6 Architecture #6: Encoding-Decoding One-Dimensional CNN (ED-1D-CNN)

1D-CNN architectures have most extensively been employed in medical time-series studies, including the analysis of ECG and EEG [171], and cardiotocography traces [172]. Russo et al. [173] apply 1D-CNN architecture with deep autoencoders for anomaly detection for wastewater monitoring systems. A survey of various applications of 1D-CNN is provided in [174]. Chollet [94] including an example software program for a 1D-CNN using the Keras neural network library. However, there are no known published papers that utilize ED-1D-CNN architecture.

CHAPTER 7 – DEEP NEURAL NETWORK (DNN) EXPERIMENTATION

7.1 Background

Six (6) neural network architectures are described in Chapter 6. Five architectures are based in part on an autoencoder (#1, #2, #3, #5, and #6), one architecture includes a generative adversarial network (#4), one architecture includes a gaussian mixture model (#3), one architecture includes a recurrent neural network (#5), and one architecture includes a convolutional neural network (#6). Architecture #1 may be viewed as two architectures; a shallow autoencoder has a single hidden layer, and a deep autoencoder has multiple hidden layers. The deep network should produce more accurate results, but the shallow network will execute faster and maybe preferable within a streaming architecture. Autoencoders are trained with feedforward networks and are algorithmic more stable compared to GANs, RNNs, and CNNs.

GANs are estimated using dual feedforward networks resulting is slower parameter training. Therefore, the deployment in streaming architectures is problematic. However, the use of GANs in anomaly detection studies is emerging, and more research is needed regarding architectural options and streamlined algorithms. The GAN in use here implements a plain vanilla design that lacks the recent advances in architectural optimizations that have emerged in the literature.

An RNN-based architecture is designed to exploit the temporal structure where samples depend on one or more previous samples. If there does not exist a temporal dependence, then the deployment of RNN models would be counterproductive, and an autoencoder or other non-recurrent architecture will provide better results. The CNN-based architecture #6 traditionally

used for image processing has been adapted for sequence processing and has the same strengths and limitations of an RNN.

The four (4) experimentation datasets are sequential with a timestamp and are candidates for the application of the RNN and 1D-CNN architectures. The fraud dataset is based on sequential transactions over time, the DATACENTER dataset is based on the logging of data center activity over time, and the SWAT and WADI datasets are based on the recording of sensors, actuators, and network devices over time. However, the temporal dependencies contained in these datasets are weak. For example, the fraud transactions were recorded sequentially but by different purchasers. Application of RNN and 1-CNN-based architectures is possible but might not yield improved results. Note that the RNN in architecture #5 utilizes the Long Short-Term Memory (LSTM) variant of the architecture for estimation stability and improved temporal dependency modeling.

7.2 t-SNE Visualization

t-SNE is a visualization technique developed by van der Maaten and Hinton [175] designed to explore high-dimensional data relationships. The goal of t-SNE is to transform a set of high-dimensional points into a two-dimensional representation in order to visualize clusters of related data. The t-SNE iterative algorithm uses SGD and performs different non-linear transformations on different regions of the data, and converts similarities between data points to joint probabilities. The algorithm tries to minimize the Kullback-Leibler (K-L) divergence (discussed in Section 6.3.1) between the joint probabilities from the low-dimensional representation and the high dimensional data. T-SNE has a set of hyperparameters such as the learning-rate and a perplexity value, usually ranging from one (1) to one-hundred (100). With smaller perplexity values, local variations dominate, while larger perplexity values the clusters

are merged based on the similarity of features. Overall, the results are influenced by the perplexity value, the number of iterations, and other parameters required by the SGD algorithm.

Figure 38 displays the t-SNE representations for the four experimentation datasets with perplexity values zero (0) and fifty (50). The plots in the left column with a t-SNE perplexity value of 0 is an unprocessed representation of the raw high-dimensional dataset transformed into a two-dimensional representation. Each point represents an anomaly in red or a non-anomaly in green. With a perplexity value of 0, the anomalies cannot be distinguished from the non-anomalous samples. The inability to distinguish anomalies from non-anomalies indicates that the raw data are evenly distributed and has the appearance of a Gaussian distribution.

The plots in the right column of Figure 38 display the datasets transformed with a perplexity value of fifty (50). Improved separation of the data is shown with apparent clustering and with the red anomaly points partially separated from the green points. The SWAT and WADI datasets have the best separation with the fraud dataset also exhibits a few densely populated red clusters. Conversely, the DATACENTER dataset shows less separation, which is an indication that the limited number of features, seven (7) in total, is insufficient for the t-SNE algorithm to process.

Note that the topology of the t-SNE clusters is somewhat tricky to interpret. Symmetry indicates underlying Gaussian distributions. For example, the elongated shapes in the WADI and SWAT plots indicate an axis-aligned Gaussian distributed. These results also indicate which datasets are likely candidates for application of a mixture of Gaussian techniques such as architecture #3, DA-GMM. Plots that lack structured might indicate that Gaussian-based techniques are not appropriate. In these instances, consideration should be given to the application of distribution-agnostic autoencoders or distance-based GAN architectures.

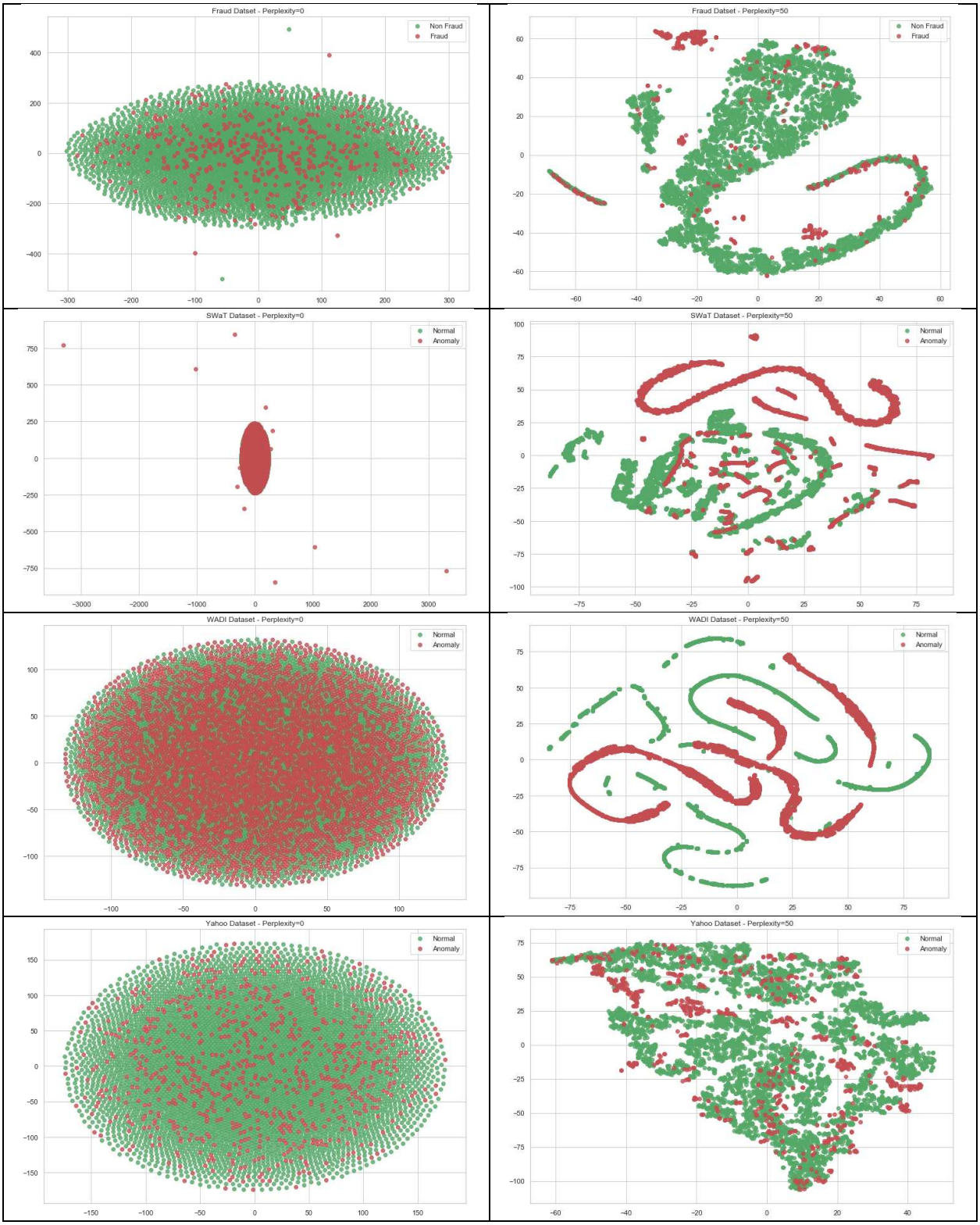


Figure 38: t-SNE Plots – 0 and 50 Perplexity

7.3 Experimentation Methodology and Hyperparameters

Experimentation was performed in unsupervised mode, meaning that the model parameters are trained without regard to the anomaly labels. Anomaly labels are used in testing but only for evaluation purposes. The FRAUD and DATACENTER training datasets contain anomalies, but in a low percentage, since, by definition, anomalies are rare events. The reason is that in real-world datasets, anomalies are rarely known or designated a priori. Note that the training samples from the SWAT and WADI testbeds were captured during the period of benign operation (no network attacks). So, unlike the FRAUD and DATACENTER training datasets, the SWAT and WADI training datasets are free of anomalous samples. The estimation challenge here is to identify anomalies in test data based on parameter training in an unsupervised environment with anomaly-free data.

All datasets are time-series and include a time-stamp attribute. The ED-RNN and ED-1D-CNN architectures require timestamps for the sample to look back; otherwise, the timestamps are ignored.

Architectures with autoencoders use feedforward network with the backpropagation algorithm, RNNs use the backpropagation-through-time (BPTT) algorithm, and the CNN use the convolution neural network algorithm. SGD with the ADAM [71] or RMSprop optimizers is used throughout the experimentation. Per standard DNN estimation practice, training occurs over multiple epochs, with each epoch containing a mini-batch that includes a subset of the complete set of training samples. Mini-batch sizes are architecture and dataset dependent; excessive epochs and mini-batch sizes result in model overfitting and poor test results.

Evaluation metrics included the area under the receiver operating characteristic (ROC) curve (AUC) and other metrics described in section 2.5. The AUC is preferred because the

calculation is based on the entire range of anomaly score thresholds. The confusion matrix, precision, recall, and F_1 metrics are point estimates derived from a single arbitrary anomaly score threshold. While not the best performance metric in the presence of highly unbalanced data, 'confusion matrix' results are nevertheless presented in addition to AUC to describe absolute performance (e.g., true positives) comparable across architectures.

Table 16 describes the anomaly scoring approaches used for each architecture. As a binary classification problem, for evaluation purposes, a threshold score or other deterministic approach is required. Four (4) different approaches are built into the architectures: (a) 'reconstruction error,' (b) 'probability of an anomaly,' (c) 'energy' with percental level,' and (d) chained DNN-TML. The chained DNN-TML is a novel approach where the latent representation output produced by the DNN VAE is inserted into the TML HBOS algorithm. Per standard practice and for efficient DNN estimation, all inputs were identically scaled and normalized with mean zero () and standard deviation one () across the entire spectrum of experimentation. This scaling means that the identical input reconstruction thresholds can be used in the autoencoding architectures #1, #5, and #6.

All experimentation algorithms were programmed using the Python-3 programming language with the Tensorflow™ deep learning library. Where possible, software implementations were validated against other similar published findings. For example, the DA-GMM implementation faithfully replicates the numeric results and graphics presented in the original paper [85].

Table 16: Architecture Anomaly Scoring Summary

Architecture	Anomaly Scoring	Threshold for Experimentation
#1 Shallow-Deep Autoencoder (SDA)	Input Reconstruction Error	Threshold > 4 (FRAUD, SWAT, WADI) Threshold > 1 (DATACENTER)
#2 Variational Autoencoder (VAE)	Ensemble DNN-TML using HBOS	Built-Into HBOS. Equal to the percentage of anomalies in the latent representation of test
#3 Deep Autoencoding Gaussian Mixture Model (DA-GMM)	Sample Energies derived by the Gaussian Mixture Model (GMM) [85]	All percentiles presented (e.g., energy Level such that 20% of samples are anomalous.
#4 Generative Adversarial Model (GAN)	Probability of a Fake	Anomaly if probability > .5 is fake as determined by the discriminator
#5 Encoding-Decoding Recurrent Neural Network (ED-RNN)	Input Reconstruction Error	Threshold > 4 (FRAUD, SWAT, WADI) Threshold > 1 (DATACENTER)
#6 Encoding-Decoding Convolutional Neural Network (ED-1D-CNN)	Input Reconstruction Error	Threshold > 4 (FRAUD, SWAT, WADI) Threshold > 1 (DATACENTER)

7.4 Architectures #1 – Shallow Deep Autoencoder (SDA) Results

Table 17 displays the training size, test size, anomaly counts, and various DNN parameters used in the SDA estimation. Datasets were split with seventy (70) percent for training and thirty (30) percent for the test. As a feature extractor, autoencoders are designed with fewer hidden units than the number of input features. For instance, the SWAT shallow autoencoder model included a single hidden layer of twenty-five (25) nodes, resulting in 2320 trainable parameters. The corresponding deep autoencoder model has four hidden layers of 13-7-7-13 nodes resulting in 3384 trainable parameters. Note the symmetry in the number of DNN layers between the encoder and decoder components. The encoder includes a succession of decreasing number of nodes, while the decoder includes a succession of an increasing number of nodes. All estimation utilized rectified linear units (RELU) as the nonlinear activation function,

the Adam SGD optimizer, an SGD learning rate of 1e-7, fifty (50) training epochs, with each epoch utilizing an SGD batch size of 128.

Table 17: Shallow-Deep Autoencoder DNN Parameters

Dataset	Training Size	Test Size	Anomalies in Test	Input	Hidden Layers	Output	Trainable Parameters
FRAUD (Shallow)	199364	85443	157	30	15	30	945
SWAT (Shallow)	495000	449919	54621	45	25	45	2320
WADI (Shallow)	1209601	172801	9860	99	60	99	12039
DATACENTER (Shallow)	117600	50400	246	8	4	8	76
FRAUD (Deep)	199364	85443	157	30	18-10-6-6-10-18	30	1694
SWAT (Deep)	495000	449919	54621	45	26-13-7-7-13-26	45	3384
WADI (Deep)	1209601	172801	9860	99	60-30-15-15-30-60	99	16914
DATACENTER (Deep)	117600	50400	246	8	6-5-3-3-5-6	8	231
All Datasets	Optimizer=Adam, Activation=RELU, Learning Rate=1e-7, 50 Training Epochs=50, Batch Size=128						

7.4.1 Architecture #1: Shallow Autoencoder Results

Table 18 and Figure 39 provide the findings from the shallow autoencoder experimentation. The results indicate that shallow autoencoders are reasonably capable anomaly detectors, as evidenced by the AUC associated with the FRAUD (.961), SWAT (.878), and WADI (.783) experimentation. The shallow autoencoder performed poorly with the DATACENTER dataset.

The AUC for the FRAUD experimentation is surprisingly high, given the imbalance of the test data and the difficulty in 'needle in the haystack' detection with unsupervised techniques. There were 85443 samples and only 157 anomalies in the FRAUD test dataset, for an anomaly rate of .0018. If the anomaly scores are ranked from top to bottom, and an arbitrary threshold value of four (4) is specified, then 118 'true positives' and 39 'false negatives' are designated from the anomaly pool. Note that while 84394 'true negatives' (non-anomalies) are correctly designated, another 892 'false positives" are incorrectly designated. If the threshold value is increased, fewer true positives are identified, but also fewer false positives are identified. This trade-off between correct and incorrect designations of anomalies is intrinsic to binary

classification problem domains. Note that the temporal sequence in the FRAUD dataset is irrelevant since each transaction is autonomous.

The AUC (.878) for the SWAT (.878) experimentation and, to a lesser extent, for the WADI experimentation (.783) also illustrates reasonably good performance. The SWAT test dataset includes 449919 samples with 54621 designated anomalies by the testbed experimentation team, for a rate of 12.1 percent. The WADI test dataset included 172801 samples with 9860 inferred anomalies, for a rate of 5.7 percent. The WADI anomalies are inferred because the testbed experimentation team did not explicitly designate anomalies. An anomaly in the WADI dataset is defined broadly as the samples recorded during periods of testbed cyber or network attack. All samples acquired during these known attack periods are designed anomalous. In general, the SWAT anomaly designation approach is more accurate than the WADI approach, which may account for differences in algorithmic performance. Experimentation results will be highly dependent on the quality of the underlying data collection process and the corresponding anomaly designation assumptions.

Table 18: Shallow Autoencoder Experimentation Results

Metric	Fraud	SWAT	WADI	Datacenter
Area Under ROC	.962	.878	.783	.579
True Negative	84394	391941	157770	47090
False Positive	892	3357	5171	3086
False Negative	39	21861	6583	224
True Positive	118	32760	3277	22
Precision	.116	.907	.387	.007
Recall	.751	.599	.332	.009
F_1 Score	.202	.722	.357	.013

The fact that the AUC for the SWAT and WADI experimentation is lower than the FRAUD experimentation might be a by-product of model overfitting. The SWAT and WADI experimentation demonstrated an increase in model loss over the training epochs, a symptom of

overfitting. This conclusion is illustrated with the training and test loss curves displayed in the left column of Figure 39. While the FRAUD and DATACENTER datasets illustrate the typical convex shape of these loss curves, the SWAT and WADI loss curves are either flat-lined, trending upward, or convex. In general, the test loss curve will be above training loss and will eventually level-off, indicating that the autoencoding parameters stabilize around ten (10) epochs, and further estimation is counterproductive.

There are various DNN techniques to combat overfitting, including additional parameters for L1 and L2 regularization, random 'dropout' of nodes in hidden layers, and reducing the number of epochs, also known as 'early stopping.' Models with otherwise identical parameters might overfit one dataset and underfit another dataset and may depend on the specification of the hidden layer(s) and the complex interaction with the other hyperparameters.

The right column of Figure 39 also displays the Receiver Operating Characteristics (ROC) curve. Recall that the ROC curve measures the relationship between false positive and true positive rates; the AUC is the probability that a classifier will rank a true positive higher than a false negative. A true positive indicates that the model correctly identifies the anomaly. Since a positive is an anomaly, a false negative indicates that the model fails to identify the anomaly. Note that in many mission-critical applications, the risk is not necessarily symmetrical; the risk of failing to identify is greater than the risk to over-identifying anomalies. For these and other reasons, the ROC curve should be well above the diagonal line. An AUC higher than .75 indicates that the anomaly classifier is reasonably performant using the shallow autoencoder with the FRAUD, SWAT, and WADI datasets.

The AUC associated with the DATACENTER experimentation is .579. This result is only slightly above a random chance result and indicates that the autoencoder architecture does not identify anomalies in the DATACENTER dataset.

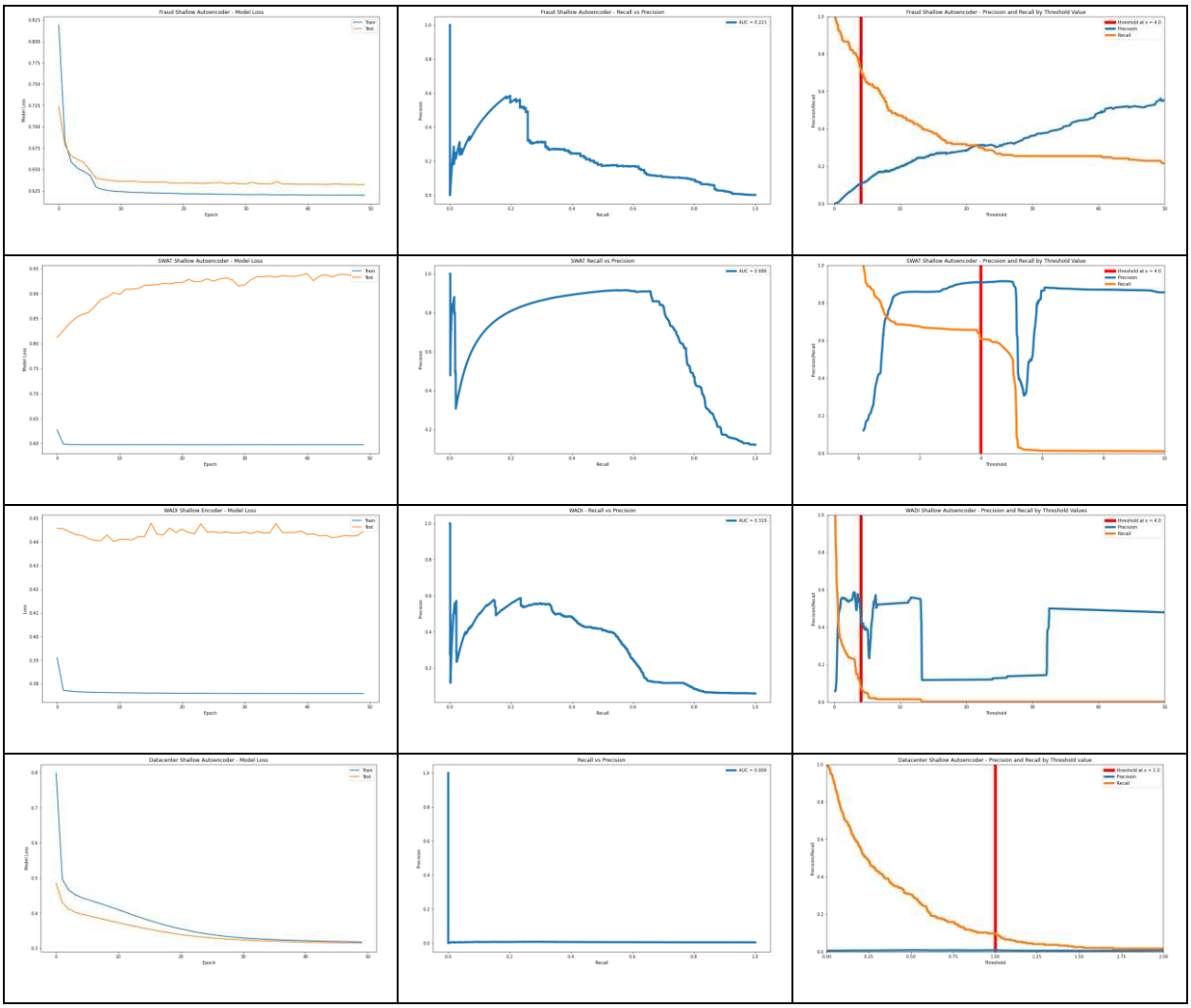


Figure 39: Shallow Autoencoder Training

7.4.2 Architecture #1: Deep Autoencoder Results

Table 19 and Figure 40 provide the findings from the deep autoencoder experimentation. The results are similar to the shallow autoencoder indicating the additional DNN layers adds a small to the performance of the autoencoder as an anomaly detector. The AUC is now .894 (compared to .878) for the SWAT experimentation and .818 (compared to .783) for the WADI

experimentation. The DATACENTER AUC of .544 is now lower than the shallow autoencoder AUC of .579. The overall conclusion can be drawn that there is a minimal performance bump, if any, from a deep over a shallow architecture.

Note that the deep autoencoder execution time (~ 12 seconds) is approximate twice the shallow autoencoder execution time (~7 seconds) but is still fast enough to be adaptable to a streaming architecture. Moreover, the overfitting of the WADI dataset still exists but is less pronounced.

Table 19: Deep Autoencoder Summary Experimentation Results

Metric	FRAUD	SWAT	WADI	DATACENTER
Area Under ROC	.958	.894	.818	.544
True Negative	84237	391883	161037	40035
False Positive	1049	3415	1904	10119
False Negative	42	21578	9196	172
True Positive	115	33043	664	74
Precision	.102	.906	.258	.007
Recall	.751	.604	.067	.300
F_1 Score	.180	.725	.106	.014

Consider the SWAT dataset with a deep autoencoder. In total, 391843 samples were correctly classified as non-anomalous, and 33043 samples were correctly classified as anomalous, for a total of 424886. However, 3415 were incorrectly classified as anomalous, and 21578 were incorrectly classified as non-anomalous, for a total of 24993 incorrect classifications. Therefore, the misclassification rate is 5.5 percent (24993 / (424886 + 24,993)). Given the imbalance of data, a naïve approach would be to classify all samples as non-anomalous. This naïve approach would produce a misclassification rate of 12.1 percent (54621 / 449919). Following Occam's razor, all things being equal, simplicity is preferred over complexity. In this example, the more complex deep autoencoder model produces superior classification results over the naïve model. Note that what is and what is not acceptable, and the anomaly threshold values

are entirely domain-dependent and sensitive to the problem requirements. A naïve model may be acceptable in one domain and unacceptable in another.

To summarize, there seems to be a slight advantage to the use of deep autoencoders over the less complicated shallow autoencoders. SDAs are not based on probability distribution assumptions, another difference from the VAE and AE-GMM models discussed in the following sections.

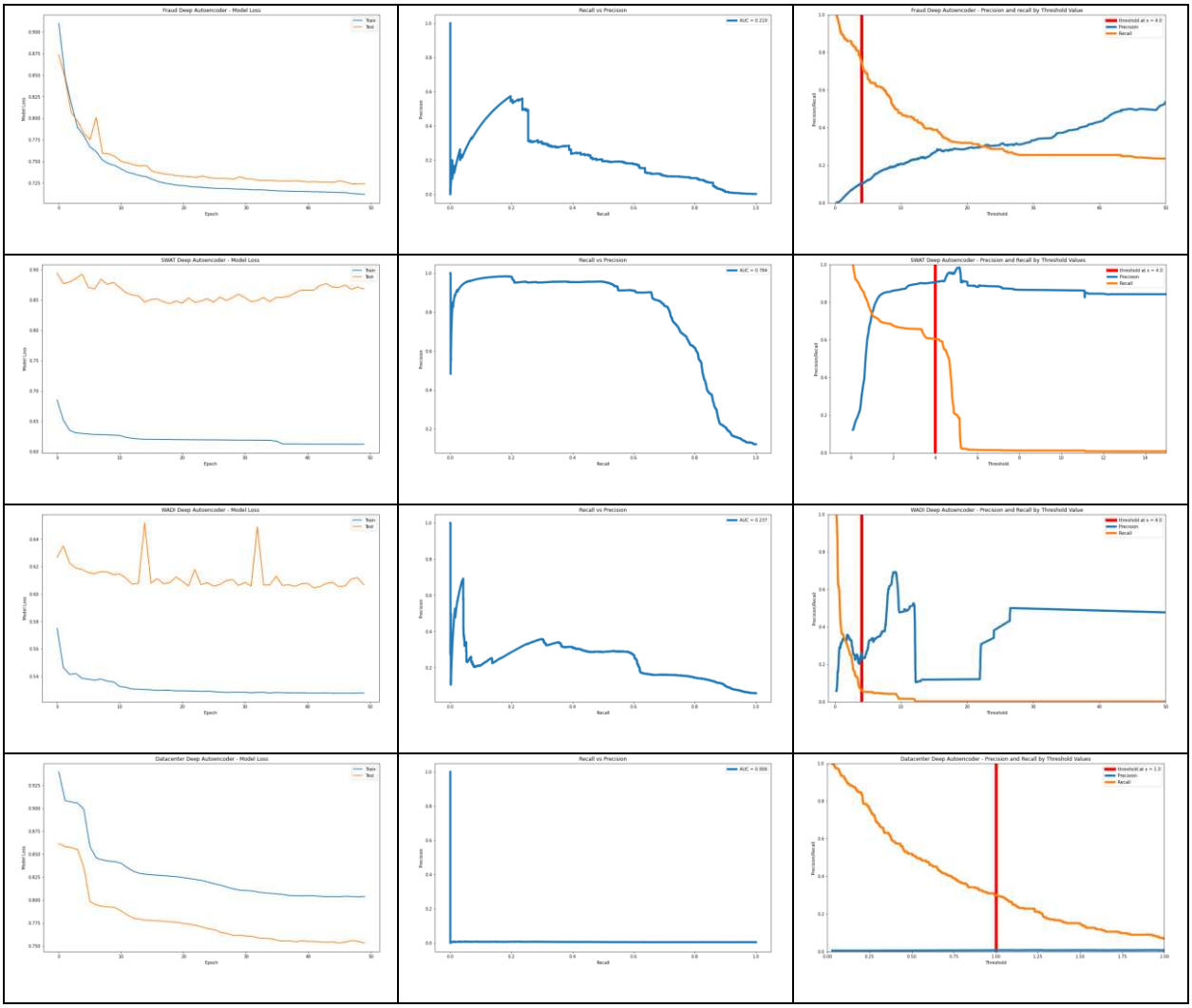


Figure 40: Deep Autoencoding Training

7.5 Architecture #2: Variational Autoencoder (VAE) Results

A VAE is an unsupervised technique made accessible in the generative image literature but adapted here for anomaly detection. A VAE consists of a DNN encoder, a DNN decoder, and a loss function. The DNN encoder compresses the inputs into a lower-dimensional space, known as the latent space, and outputs the parameters of a Gaussian probability density. The DNN decoder inputs the latent representational space and outputs the parameters to the probability distribution function. The loss function measures how effectively the decoder can reconstruct the inputs. Once these parameters are estimated from the training data, the decoder is no longer required. Under VAEs, rather than learning to replicate the input data, the autoencoder learns the parameters of the latent representation of the data.

At least two approaches are possible to adapt the VAE architecture to anomaly detection. The first approach is to apply the estimated VAE parameters to generate the latent space representation of the test. Sample points from the test latent space distribution can be randomly drawn and compared against the test samples. Anomalies can be identified when the test samples diverge measurably from the sampled latent representation of the training samples.

The second approach, adopted here, leverages the findings from the TML experimentation discussed in Chapter 5 to form a unique chained, ensemble technique. As before, as new samples are received, the parameters estimated from the training dataset are used to produce a compressed, multidimensional representation. An unsupervised TML technique is then applied to this representation to identify anomalies within the latent space. Histogram-Based Outlier Score (HBOS) is the technique selected here, but any of the unsupervised TML techniques (e.g., k-nearest neighbor) described in Chapter 4 may be used. Table 20 presents the estimation results, and Figure 41 provides the training model graphs for the VAE architecture

with HBOS. Note that the last line of Table 20 provides the number of non-trainable / trainable in the VAE model for each dataset.

Table 20: Variational Autoencoder Experimentation Results

Metric	FRAUD	SWAT	WADI	DATACENTER
Area Under ROC	.687	.783	.652	.583
True Negative	77718	373234	149867	45251
False Positive	7568	22064	13074	4903
False Negative	93	33811	5715	201
True Positive	64	20810	4145	45
Precision	.008	.485	.240	.009
Recall	.407	.380	.420	.182
F_1 Score	.016	.426	.306	.017
# Parameters	92 / 1214	122 /1214	230 / 3629	28 /184

The results from the VAE experimentation show promise but are less performant than the SDA results. The VAE AUC metric is well below the corresponding SDA AUC for the FRAUD, SWAT, and WADI datasets and slightly higher for the DATACENTER dataset. The AUC metric dropped precipitously from .958 with SDA to .687 with VAE. Note that the WADI VAE produced a more substantial true positive value than WADI SDA experimentation (4145 versus 664) but at the expense of a more substantial false positive value (13074 versus 1904).

Because of the probabilistic nature of a VAE, no two experimentation runs of the model will produce identical results unless the entire sequence of Gaussian random draws from the latent representation is controlled. The model loss curves in Figure 41 are convex and demonstrate that overfitting is not an issue. Otherwise, the shapes of the loss curves are comparable to the SDA experimentation. While the VAE technique described here produced less optimal results, the underpinnings of VAE are more theoretically justifiable than non-probabilistic autoencoders. Further research is needed concerning the exploitation of latent space representations and the chaining of DNN with TML algorithms for anomaly detection.

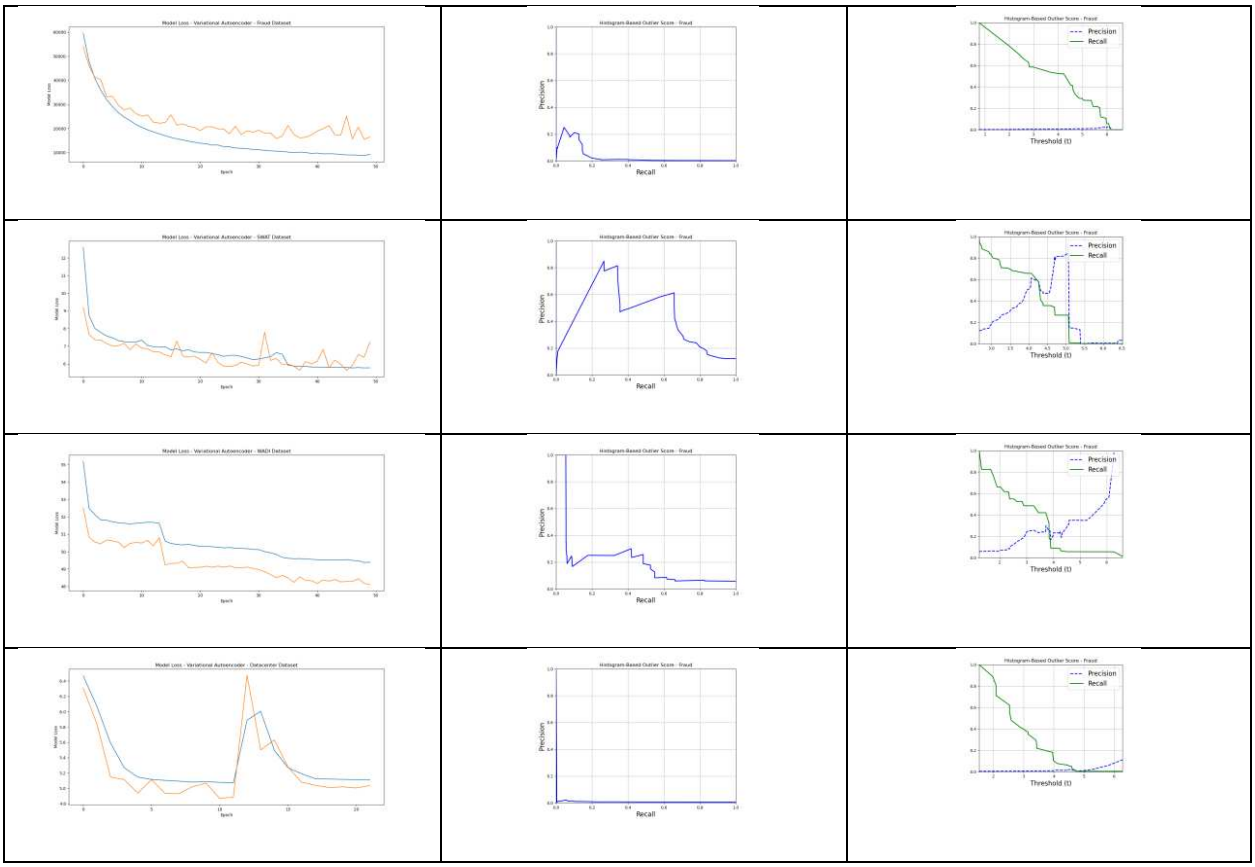


Figure 41: Variational Autoencoder Training

7.6 Architecture #3: Deep Autoencoding Gaussian Mixture Model (DA-GMM) Results

The DA-GMM algorithm is programmatically more complex than the SDA or the VAE, and the complete software implementation details are beyond the scope here. The DA-GMM approach is to use a deep autoencoder that produces a reconstruction error for the inputs, which is fed into a Gaussian Mixture Model. The algorithm jointly optimizes the parameters of the autoencoder with the mixture model simultaneously. With the test dataset, using the learned GMM parameters, sample energies are predicted; the higher the energy, the more likelihood of an anomaly. The concept of 'energy' is defined using an expression for the multivariate Gaussian distribution that involves a covariance matrix inversion. In practice, there are often matrix inversion and performance issues, so the implementation uses the Cholesky decomposition of the covariance matrix as a substitute for matrix inversion.

Figure 42 displays a set of graphics derived from the training of the DA-GMM model. Total loss, reconstruction loss, sample energy, and the value of the covariance diagonal over the training cycles are displayed for each dataset. Note that unlike other approaches, the DA-GMM curves lack smoothness and exhibit sharp changepoints and will require more epochs to stabilize the estimated parameters. The number of training epochs was set at 300, resulting in longer execution times than either the SDA or VAE architectures.

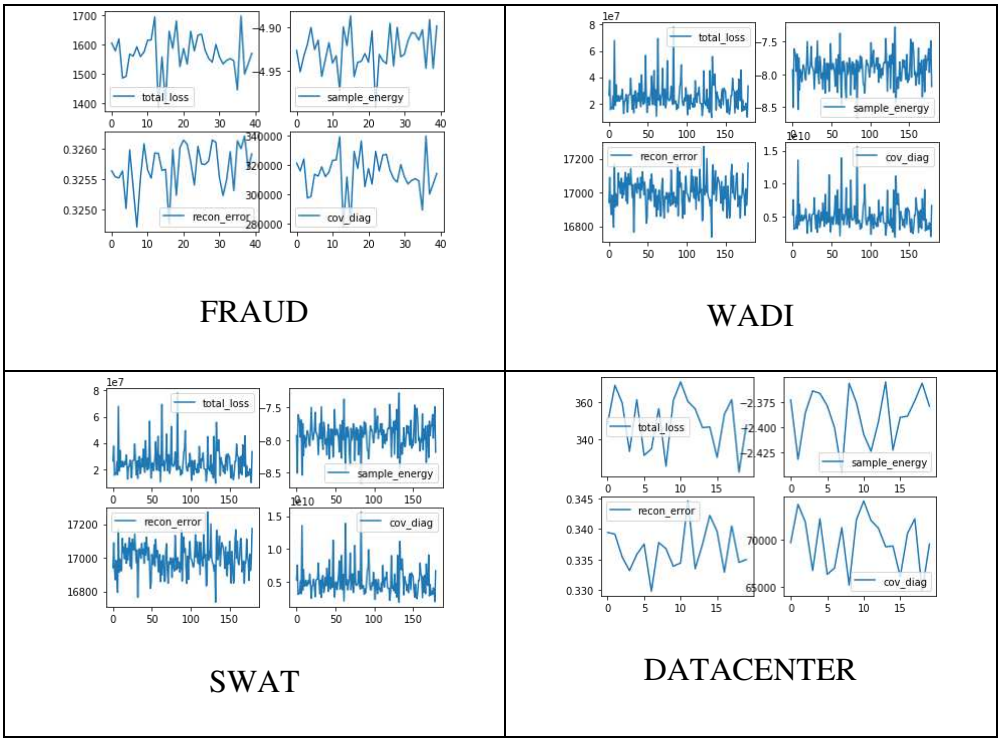


Figure 42: Deep Autoencoding Gaussian Mixture Model Training

Tables 21-24 displays the results of estimation covering the full range of sample energy percentiles ranging from 0 to 100 percent. Note that when the energy percentile is set to zero (0), all samples are considered anomalies. Conversely, when the energy percentile is set to one, all samples are considered non-anomalous. The higher the energy threshold, the fewer of designated anomalies.

Table 21: Deep Autoencoding Gaussian Mixture Model Experimentation – FRAUD

	Percentile of Energies												
	0	**5**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
Area Under ROC	.500	.518	.543	.587	.631	.671	.718	.762	.812	.859	.874	.628	.500
True Negative	0	4270	8543	17085	25627	34168	42711	51254	59798	68341	76874	81061	85285
False Positive	85286	81016	76743	68201	59659	51118	42575	34032	25488	16945	8412	4225	1
False Negative	0	2	2	4	6	9	10	12	12	13	24	109	157
True Positive	157	155	155	153	151	148	147	145	145	144	133	48	0
Precision	.002	.002	.002	.002	.003	.003	.003	.004	.006	.008	.016	.011	.000
Recall	1.00	.987	.987	.975	.962	.943	.936	.924	.924	.917	.847	.306	.000
F_1 Score	.004	.004	.004	.004	.005	.006	.007	.008	.011	.017	.031	.022	.000
Energy Threshold	-5.34	-5.22	-5.18	-5.09	-4.96	-4.79	-4.59	-4.37	-4.12	-3.61	-1.33	1.05	45.51

Table 22: Deep Autoencoding Gaussian Mixture Model Experimentation – SWAT

	Percentile of Energies												
	0	**5**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
Area Under ROC	.500	.522	.547	.588	.628	.661	.688	.721	.606	.443	.467	.484	.500
True Negative	0	21896	44110	87525	130887	173614	215706	258410	286970	310857	352609	374088	395298
False Positive	395298	373402	351188	307773	264441	221684	179592	136888	108328	84441	42689	21210	0
False Negative	0	600	882	2459	4089	6354	9253	11541	27973	49078	52318	53335	54620
True Positive	54621	54021	53739	52162	50532	48267	45368	43080	26648	5543	2303	1286	1
Precision	.121	.126	.133	.145	.160	.179	.202	.239	0.197	0.062	.051	.057	1.00
Recall	1.00	.989	.984	.955	.925	.884	.831	.789	.488	.101	.042	.024	0.00
F_1 Score	.217	.224	.234	.252	.273	.297	.325	.367	.281	.077	.046	.033	0.00
Energy Threshold	-4.67	-2.97	-1.00	4.63	4.98	5.54	6.92	7.21	8.94	9.76	11.303	13.16	606.52

The DA-GMM model performed poorly on all four datasets. The highest AUC for the FRAUD dataset (.874) occurred at the 80th energy percentile; for the SWAT dataset (.721) at the 60th percentile; for the WADI dataset (.511) at the 10th energy percentile, and the DATACENTER (.524) at the 70th energy percentile. These results are all significantly below the AUCs found with the SDA and VAE architectures. The highest AUC in the FRAUD dataset resulted in 133 true positives and only 24 false negatives being identified but at the expense of

creating 8412 false positives. The reasons for this poor performance are not entirely clear. The AE-GMM architecture is more theoretically grounded than the SDA architecture, but the core underlying Gaussian assumptions may be inappropriate for these datasets.

Table 23: Deep Autoencoding Gaussian Mixture Model Experimentation – WADI

	Percentile of Energies												
	0	**5**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
Area Under ROC	.500	.515	.511	.509	.468	.431	.384	.430	.465	.507	.560	.508	.500
True Negative	0	8437	16504	32772	48295	63905	79321	96476	113410	130490	147760	154959	162940
False Positive	162941	154504	146437	130169	114646	4649	83620	66465	49531	32451	15181	7982	1
False Negative	0	203	771	1788	3544	5211	7078	7202	7543	7749	7757	9200	9860
True Positive	9860	9657	9080	8072	6316	4649	2782	2658	2317	2111	2103	660	0
Precision	.057	.059	.058	.058	.052	.045	.032	.038	.045	.061	.122	.076	.000
Recall	1.00	.979	.922	.819	.641	.472	.282	.270	.235	.214	.213	.067	.000
F_1 Score	.108	.111	.110	.109	.097	.082	.058	.067	.075	.095	.155	.071	.000
Energy Threshold	-3.19	-2.96	-2.76	-2.22	-1.18	4.63	5.29	5.30	5.31	5.33	5.51	5.88	145.59

Table 24: Deep Autoencoding Gaussian Mixture Model Experimentation – DATACENTER

	Percentile of Energies												
	0	**5**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
Area Under ROC	.500	.496	.501	.482	.483	.502	.506	.521	.524	.511	.508	.495	.500
True Negative	0	2506	5016	10022	15038	20063	25080	30103	35120	40129	45143	47644	50153
False Positive	50154	47648	45138	40132	35116	30091	25074	20051	15034	10025	5011	2510	1
False Negative	0	14	24	58	82	97	120	137	160	191	217	236	246
True Positive	246	232	222	188	164	149	126	109	86	55	29	10	0
Precision	.005	.005	.005	.005	.005	.005	.005	.005	.006	.005	.006	.004	.000
Recall	1.00	.943	.902	.764	.667	.606	.512	.443	.350	.224	.118	.041	.000
F_1 Score	0.010	.010	.010	.009	.009	.010	.010	.011	.011	.011	.011	.007	.000
Energy Threshold	-3.72	-3.66	-3.61	-3.50	-3.39	-3.27	-3.16	-2.95	-2.66	-2.12	-0.98	0.74	583.3

7.7 Architecture #4: Generative Adversarial Network (GAN) Results

Table 25 displays the experimentation results produced from the GAN architecture. The AUC results are on-par with the DA-GMM architecture, with the best results demonstrated by

the FRAUD dataset (AUC=.872) and the worst results experienced by the DATACENTER dataset (AUC=.502). Note that the F_1 scores using the GAN architecture are higher than the F_1 scores using the DA-GMM architecture. Recall that the AUC is a summary metric encompassing all of the threshold values, while the F_1 score is based at a given anomaly threshold level. The F_1 score is .753 for FRAUD is and .744 for SWAT. So, to summarize, a GAN makes no explicit assumptions regarding underlying distributions, produces relatively accurate results, but exhibit longer execution times.

Table 25: Generative Adversarial Network Experimentation Results

Metric	FRAUD	SWAT	WADI	DATACENTER
Area Under ROC	.872	.800	.657	.502
True Negative	85250	394535	160121	49832
False Positive	36	763	2820	322
False Negative	40	21750	6585	243
True Positive	117	32871	3275	3
Precision	.764	.977	.537	.009
Recall	.745	.601	.332	.012
F_1 Score	.754	.744	.410	.010
# Parameters	158 / 3341	158 / 4301	158 / 7757	158 / 1933

Figure 27 provides a new-style graphic of the training performance of the GAN with respect to each of the four datasets. These graphs display the value of the F_1 metric on the test or dataset as training proceeds. Note the up-and-down flow of the graph until training stabilizes, and the graph flatlines. With adversarial training, two neural networks run concurrently and require more epochs (e.g., 5000) than autoencoders to sufficiently train the parameters. A GAN requires longer execution times, roughly ten times the execution time of the deep autoencoding architecture.

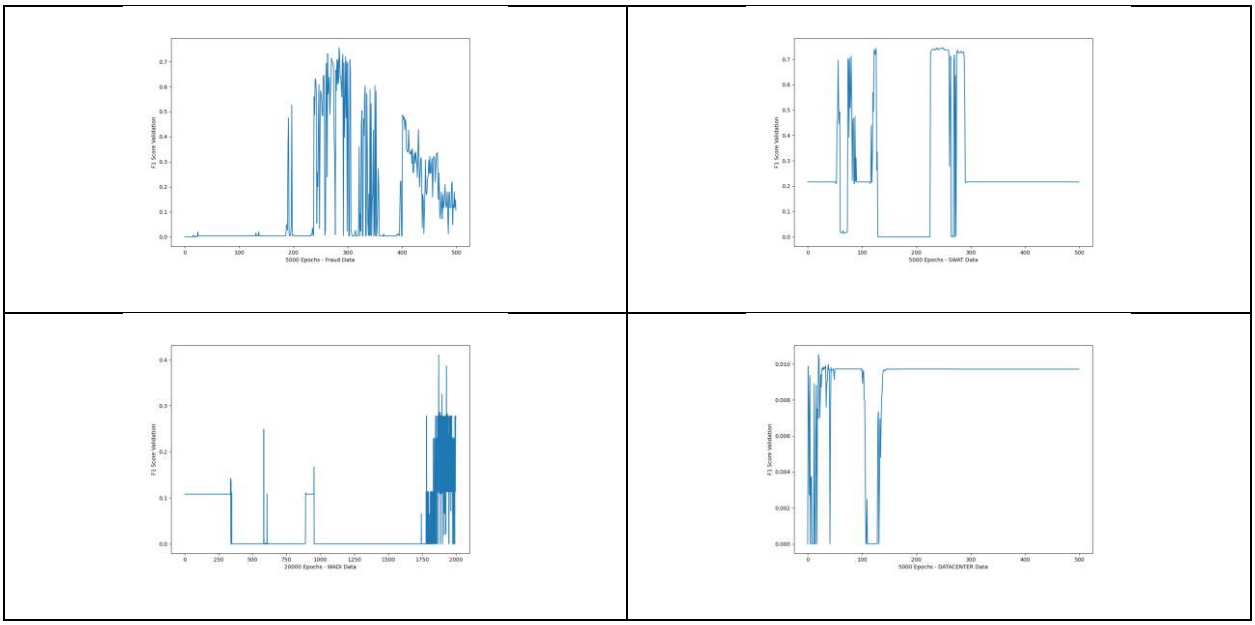


Figure 43: Generative Adversarial Network Training

7.8 Architecture #5: Encoder-Decoder Recurrent Neural Network (ED-RNN) Results

The final two architectures under consideration, ED-RNN, and ED-1D-CNN incorporate a look-back feature. The ED-RNN architecture is essentially a recurrent neural network that acts like an autoencoder. This architecture explicitly models temporal relationships and long-term memory, which is essential when considering spatiotemporal applications. While RNNs have been used extensively for sequence forecasting, neural machine translation, and other sequence-to-sequence models, they have been infrequently used for anomaly detection applications. Using the ED-RNN, the encoder and decoder are RNN. Similar to architecture #1, the shallow and deep autoencoder, the input reconstruction error is the measure of the anomaly; when the reconstruction error exceeds a pre-defined threshold value, an anomaly is designated. Because the scales of the input are identical, the threshold values are also the same as architecture #1.

Table 26 presents the experimentation results for the ED-RNN architecture for the FRAUD dataset; Table 27 for SWAT; table 28 for WADI; and Table 29 for DATACENTER.

Four different ‘look-back’ periods are included: five (5), ten (10), twenty-five (25), and fifty (50) with a period typically one-second in length. A comparison with the architecture #1 deep autoencoder results found in Table 19 is a good indicator of the value-added from the inclusion of a recurrent network in the architecture.

The AUC (.956) results of the ED-RNN FRAUD experimentation with a look back of five (5) is identical to the corresponding AUC (.958) from the deep autoencoding experimentation. This result is to be expected since the FRAUD samples are not temporally dependent. Noteworthy is the fact that the AUC (.937) is higher than the corresponding AUC (.894) in the five (5) period SWAT deep autoencoding experimentation; similarly, the AUC (.835) is higher than the corresponding AUC (.818) in the five (5) period WADI deep autoencoding experimentation. Both the SWAT and WADI datasets exhibit intrinsic temporal dependence because the network and cyber testbed attacks are multi-period. The addition of the recurrent architecture to the autoencoding architecture for the SWAT and WADI added valued and improved the anomaly detection results. Note that there was no improvement over random chance (AUC ~ .5) of the addition of recurrence to the DATACENTER experimentation results;

RNN performance is notoriously sensitive to the length of the lookback period and the combination of DNN hyperparameters. For example, the WADI experimentation failed to complete due to ‘out-of-memory’ errors experienced with a fifty (50) look back. No attempt was made to optimize performance and search for the best combination of hyperparameters.

Table 26: ED-RNN FRAUD Experimentation Results

	Look Back Period			
	5	**10**	**25**	**50**
Area Under ROC	.956	.943	.951	.933
True Negative	84064	83871	83536	82493
False Positive	1244	1426	1749	2758
False Negative	45	38	29	39
True Positive	88	104	121	137
Precision	.066	.067	.064	.047
Recall	.661	.732	.806	.778
F_1 Score	.120	.124	.119	.089
# Parameters	20574	20574	20574	20574

Table 27: ED-RNN SWAT Experimentation Results

	Look Back Period			
	5	**10**	**25 (Attack Period Only)**	**50 (Attack Period Only)**
Area Under ROC	.933	.937	.922	.896
True Negative	267037	267023	262988	253073
False Positive	66	127	4110	14004
False Negative	15461	15030	4933	4535
True Positive	910	1293	11437	11853
Precision	.932	.910	.735	.458
Recall	.055	.079	.698	.723
F_1 Score	.104	.145	.716	.561
# Parameters	40634	40634	40634	40634

Table 28: ED-RNN WADI Experimentation Results

	Look Back Period			
	5	**10**	**25 (Attack Period Only)**	**35 (Attack Period Only)**
Area Under ROC	.835	.821	.846	.861
True Negative	409290	406405	39423	35241
False Positive	2426	5336	9435	13634
False Negative	1980	1782	897	515
True Positive	1023	1185	2078	2440
Precision	.296	.182	.180	.151
Recall	.340	.401	.698	.825
F_1 Score	.317	.251	.286	.256
# Parameters	198304	198304	198304	198304

Table 29: ED-RNN DATACENTER Experimentation Results

	Look Back Period			
	5	**10**	**25**	**50**
Area Under ROC	.538	.549	.582	.527
True Negative	43596	44600	44078	38949
False Positive	6552	5531	6029	11209
False Negative	205	220	227	161
True Positive	46	46	59	66
Precision	.006	.008	.009	.005
Recall	.183	.172	.206	.290
F_1 Score	.013	.015	.018	.011
# Parameters	1384	1384	1384	1384

7.9 Architecture #6: Encoding-Decoding 1-D Convolutional Network (ED-1D-CNN) Results

The ED-1D-CNN architecture is a derivative of the ED-RNN architecture by the inclusion of a CNN layer as a preprocessing step in front of the ED-RNN architecture. CNNs are not sensitive to the order of the timestamps except within the size of the convolutional window. By the inclusion of a CNN layer, the processing is more efficient because CNN converts the long sequence into a shorter sequence that, in turn, is processed by the RNN. This approach is described in Chollet [94] in the context of a forecasting domain. The difference here is that instead of a forecast, architecture #6 is designed for anomaly detection with an autoencoder. The expectation is that the results will perhaps not be as good as architecture #5 because the approach is down-sampling from a longer time-series into a shorter CNN representation.

Table 30 displays the results from ED-1D-CNN experimentation. Consider the five (5) period look back results from the ED-RNN architecture. The results are comparable to the results from the ED-RNN. For instance, the FRAUD experimentation produced an AUC of .956 under the ED-RNN architecture and .947 under the ED-1D-CNN architecture. Similar small differences exist throughout all datasets. Therefore, the inclusion of a CNN front-end to the ED-RNN architecture has no tangible impacts; however, processing requirements are reduced.

Table 30: ED-1D-CNN Experimentation Results

Metric	FRAUD	SWAT	WADI	DATACENTER
Area Under ROC	.947	.940	.819	.534
True Negative	83622	266334	406706	41316
False Positive	1675	769	5010	8832
False Negative	33	5016	1780	196
True Positive	109	11355	1223	55
Precision	.061	.936	.196	.534
Recall	.767	.693	.407	.219
F_1	.113	.796	.264	.012
# Parameters	25662	45526	187644	3464

7.10 Experimentation Results Summary

The experimentation results from six different anomaly detection architectures were presented using four different datasets. All architectures are unsupervised, four of the architectures (SDA, VAE, ED-RNN, and ED-1D-CNN) include autoencoders, two of the architectures (VAE and DA-GMM) incorporate statistical distribution theory, two of the architectures include generative models (VAE and GAN) and two specifically model time sequences (ED-RNN and ED-1D-CNN). None of the architectures were explicitly designed for geospatial data, although CNNs can be modeled as a geospatial problem, and none of the architectures were designed for support streaming data. The only architecture ED-RNN) designed for temporal applications (ED-RNN) also has the most stringent resource requirements. The discussion of a suitable architecture of spatiotemporal streaming applications is deferred to the next chapter when the STADE architecture is presented.

Overall, the three autoencoding architectures are the most performant, followed by the GAN, VAE, and AE-GMM. The FRAUD dataset consistently exhibited the highest AUC scores, followed by SWAT, then WADI, and finally, DATACENTER. In general, all architectures performed consistently across the set of datasets in terms of accuracy across any given dataset.

Architecture performance is highly dependent on the quality of the training datasets and the anomaly labeling process. Anomaly labels are only used for architecture testing. The FRAUD and SWAT datasets were both labeled based on a set of specific rationale. For example, a sample has labeled an anomaly when a credit card transaction is denied. The testbed domain experts labeled the SWAT dataset. A WADI record was labeled as an anomaly if the sample existed during a testbed cyber-attack event. An anomaly is not determined by the recorded value

of the particular feature. This broad approach to anomaly labeling is one possible cause for the lower quality results associated with the WADI dataset.

The following are the summary key findings from the experimentation discussed in this chapter.

- All architectures performed roughly equivalently in terms of AUC across architectures.
- Shallow Autoencoders performed nearly as well as Deep Autoencoders, are simpler to implement and execute faster.
- The statistically-based approaches, VAE and DA-GMM, while more theoretically justifiable than other architectures, did not produce superior results. DA-GMM is the most complex architecture, somewhat challenging to implement, and relies on matrix inversion.
- The autoencoding recurrent neural network architecture (AE-RNN), is more justifiable when used in conjunction with temporal or sequential data, did not produce higher quality anomaly classifications. The AE-RNN architecture trains slower than other architectures by order of magnitude and would not be suitable for implementation as-is within a streaming architecture.
- The ED-1D-CNN architecture performed at the same level of AE-RNN and executes faster and is more stable.
- The Generative Adversarial Network (GAN) is easy to implement and does not rely on statistical assumptions. Training execution speed is longer than other techniques because a GAN requires two concurrent executing neural networks. Note that in the image recognition domains, GANs generate higher quality images than VAEs and are the preferred architecture.

- Anomaly detection training and testing dataset quality are problematic. Ture multivariate time-series or sequence anomaly detection benchmarks do not exist. There does not exist agreed-upon criteria to evaluate unsupervised models.
- The DATACENTER dataset, while advertised for anomaly detection research by YAHOO, is flawed and is not suitable for experimentation
- Architectures considered here do not adequately support spatiotemporal and streaming data either individually or jointly.

CHAPTER 8 – SPATIOTEMPORAL ANOMALY DETECTION ENVIRONMENT (STADE)

8.1 Introduction

Chapters 4-7 presents several unsupervised TML and DNN architectures and algorithms applied to anomaly detection. While there are many different technical options, direct architectural and algorithmic support for streaming spatiotemporal data is lacking. This support is the topic of this chapter with the specification called the Spatiotemporal Anomaly Detection Environment (STADE). STADE consists of architecture and one or more instantiations. The STADE architecture consists of a set of software or system components that are loosely connected and communicate with each other. These software components may be special-purpose unique to a particular domain or general-purpose (e.g., database system).

The mapping of the STADE software or system components to physical software or products is called an instantiation of STADE. There are multiple instantiations of STADE with each instantiation tailored to the target domain of interest. Three (3) instantiations mapped to three (3) case studies are presented in Chapters 9-11. However, most of the components of STADE are shared across the case studies.

STADE provides an environment for automated, domain-independent, globally distributed anomaly detection of multivariate streaming data. The term STADE is, coincidentally, a Greek unit of measurement, the distance covered in ancient Greco footraces. This analogy is appropriate since streaming, real-time anomaly detection, is also a race – a race against time before a decision becomes 'overcome-by-events.' With streaming applications, time is of the essence, and decisions are perishable.

8.2 Design Considerations

High-level STADE design considerations can be categorized into: (1) System Engineering, (2) Global Distribution, (3) Stream Processing, (4) Algorithmic Design, and (5) Decision Support. These categories are briefly discussed below. Note that the focus of the research is on anomaly detection algorithms for streaming spatiotemporal data and not on general computer science-related topics. However, algorithms exist in software and are controlled by an infrastructure. This infrastructure impacts not only the algorithmic design but also implementation details such as persistence, storage and retrieval, local and network communication, loose versus tight coupling, and user-interface design.

8.2.1 Systems Engineering

There are several systems of engineering performance and quality concepts that should be embraced by STADE. Examples include modularity (i.e., pluggable algorithms), responsiveness, reliability (i.e., automated recovery), spatial independence (i.e., a failure at one location does not impact the operations at another location), predictability (i.e., correctness and consistency of output), and low latency. Systems engineering textbooks describe these design, quality, and performance concepts [176].

Early prototyping is a systems engineering process for building a functional model that elicits clarity in the requirements and design before full development and operational fielding. The three case studies are examples of early prototypes. Early prototypes can demonstrate potential feasibility, scalability, and performance options and bottlenecks. For example, a prototype of a global air traffic monitoring and anomaly detection system can identify unique performance and user-interface issues that would be quite different from a social network sentiment information system.

8.2.2 Geospatial Distribution

Geospatial distribution and optimization of systems, network communications, and latency is a heavily studied and largely solved research area in computer science. Standards-based network protocols such as Transmission Control Protocol/Internet Protocol (TCP/IP), Hypertext Transport Protocol (HTTP), Message Passing Interface (MPI), and various open-source service buses support reliable and timely wide-area and cloud-based data distribution. Some of the protocols are synchronous, and some are asynchronous. Synchronous protocols have higher performance but lower reliability than asynchronous protocols.

The STADE internal architecture relies on the publish-and-subscribe messaging protocol for communication. In the publish-and-subscribe pattern, STADE sites act as publishers and subscribers of anomaly scores with other STADE sites. A messaging technique such as publish-and-subscribe, however, is asynchronous; time-critical STADE instantiations may also require point-to-point synchronous communication to satisfy requirements. Moreover, in practice, impediments to network traffic such as encryption, security devices, and firewalls also induce external network constraints and negative performance impacts.

8.2.3 Stream Processing

There are many core requirements for stream processing architectures [177]. Requirements include (a) the processing of the data in-stream upon receipt, (b) the capability to resiliently handle stream imperfections, and (c) the ability to scale resources when needed to achieve an instantaneous response. Each of these core requirements is discussed below.

Real-time management of incoming data is an essential characteristic of stream processing architectures. In-stream means that the data is processed without the employment of a persistent data store such as a relational database or file system. If data is persistently stored,

streamed and stored data should be processed concurrently without impact on the overall performance of the anomaly detection algorithm.

The ability of resiliently process stream imperfections and noise is also critical. Stream imperfections include missing, out-of-sequence, corrupted, and invalid data. Moreover, data integrity should be maintained throughout the chain of custody. Data integrity means that while data is in motion, inadvertent or malicious changes to the data should be protected. The injection of a computer virus and malware into the data are examples of the loss of integrity of a data stream.

Availability and scalability are additional streams processing requirements. Availability is the proportion of time a system is operational. Scalability is the capability to add resources and increase workload without performance degrading. Scaling-out (horizontal scaling) occurs by increasing the number of sites, while scaling-up (vertical scaling) occurs by increasing the processing resources at a given site. There are many techniques to maintain availability and scalability, including middleware, partitioning an application across multiple processors, partitioning across multiple compute nodes in a cluster and virtual machine replication on demand.

8.2.4 Algorithmic

STADE estimation algorithms are based on stochastic gradient descent (SGD) and the emerging DNN concept of federated learning (FL). With streaming data, algorithms should be online, performant, and supportive of the goals of the decision-maker. The selection of the algorithms is dependent on the selection of the Stream Anomaly Detector (SAD) and the Federated Anomaly Detector (FAD). Section 8.5.1 below describes the STADE algorithms in more detail.

8.2.5 Decision Support System (DSS)

A DSS is a digital, model-driven information system that processes and displays anomaly information and assists with decision-making. This DSS will likely require remote access as STADE sites may be situated at the compute edge or at disadvantaged network locations. The design of a STADE DSS will also be highly contextual and domain-dependent. Decision support is a complicated and broad topic that is beyond the scope of research here but is included in the architectural drawings for completeness.

8.3 Concept of Operations (CONOPS)

A CONOPS is an engineering document that describes the system's high-level architecture and interaction from an end-user perspective. The end-user may be a person, a user interface, or another digital system. STADE provides alerts to the DSS regarding the presence or absence of anomalies in streaming data. A STADE instantiation consists of the set of autonomous STADE sites that process incoming high-dimensional location or region-specific data. Each STADE site includes a plug-in capability that supports a replaceable and interchangeable algorithm called the Stream Anomaly Detector (SAD). Multiple SADs can be installed at a given site for ensemble or consensus-based anomaly score generation. Site-specific anomaly scores are pushed asynchronously through the publish-and-subscribe message bus for processing and storage at the Federated Anomaly Detector (FAD) global repository. The FAD is the central warehouse for all SAD scores within a STADE instantiation.

The site that hosts the FAD global repository site also performs anomaly detection on the streaming anomaly scores. The FAD detects ‘anomalies within anomalies’ and publishes those scores via the publish-and-subscribe message bus to the DSS. Participant sites also subscribe to global FAD scores originating from neighboring sites and may incorporate those scores as inputs

into their SAD. This architecture provides a feedback loop between site anomalies, anomalies reported from neighboring sites, and anomalies reported from the FAD global repository. Figure 44 provides a graphical view of the top-level architecture of a STADE instantiation. In this example, there are four (4) STADE geographic sites, each with a unique streaming feed (i.e., Stream #101, #102, #103, and #104).

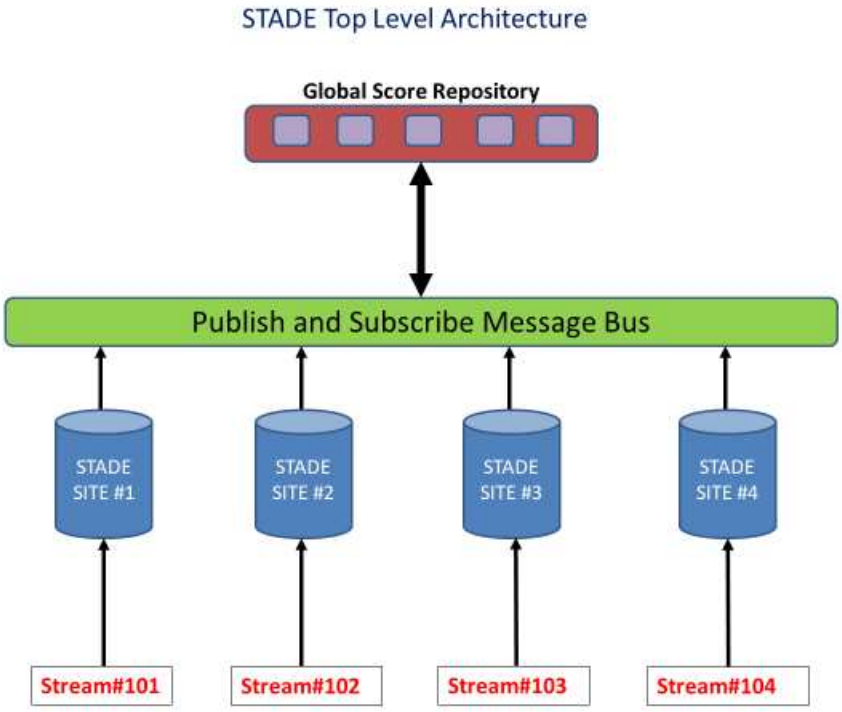


Figure 44: STADE Top-Level Architecture

8.3.1 An Aside on Federated Learning (FL)

Federated learning (FL) [178] is an emerging architecture designed to support a large number of geographically distributed networked clients. These clients aggregate into a union that collaboratively and cooperatively train DNN models using a centralized server but without input data sharing. By forming a union, robust, highly performant ML models can be estimated, modified, and deployed in real-time. FL is a form of privacy-preserving decentralized collaborative machine learning [179].

The genesis of FL originated with the need for ML on a massive number of low-powered devices such as smartphones. For example, Google™ uses FL to create smartphone keyboard assistants by training DNNs with on-device stochastic gradient descent (SGD) using captured keystrokes in real-time. However, for on-device privacy reasons, these keystrokes are not shared with a central server; what is shared periodically are the estimated DNN parameters. Even if privacy is not a concern, mobile or low-powered devices are randomly offline or have limited communication bandwidth to make centralized DNN model training impractical. By sharing model parameters and not input data, network bandwidth is preserved.

Periodically, a central FL server collects parameters from participant devices and modifies the global parameters through a technique known as Federated Averaging (FA). Under FA, the DNN parameters are aggregated and averaged across all other available and reporting devices. The newly modified SGD parameters are then transmitted back to the local phone or device. The server may add to the data stream additional privacy preservation techniques such as lossy compression for communication efficiency, update clipping, or intentional noise insertion.

There is a substantial similarity between the architectures of FL and FAD. While the privacy-preserving aspects of FL technology are not the focus of STADE, the distributed data and model parameter sharing requirements are similar in many respects. FL is designed to support a large number of unreliable devices (e.g., smartphones); STADE is designed to support a relatively small number of reliable clients. These STADE sites may be sensors with low power consumption and minimal connectivity but highly reliable.

Table 31, adapted from Kairouz et al. [180], presents a summary of the differences between the characteristics and assumptions of FL versus STADE. FL consists of hundreds or thousands of unreliable devices or sites, while STADE is designed for perhaps two (2) to ninety-

nine (99) reliable devices or sites. All sites in FL and STADE are stateful, meaning that data is store locally and persistently. FL is orchestrated and controlled by a central server while STADE uses a message bus and asynchronous communications. FL is designed to share the global DNN model parameters through FA; STADE is designed to share the anomaly scores and estimating 'anomalies within anomalies' for spatiotemporal analysis.

Table 31: Federated Learning and STADE

	Federated Learning (FL)	STADE
# of Clients	Hundreds / Thousands	1-99
Data Stored at Client	Yes	Yes
Orchestration	Centrally at Server	Publish-and-Subscribe Message Bus
Expected Client Reliability	Low Reliability	High Reliability
Network	Mobile Device	Connected / Cloud
ML Algorithm	Stochastic Gradient Descent (SGD)	Stochastic Gradient Descent (SGD)
Federated Data	SGD Model Parameters	Anomaly Scores
Model Aggregation	SGD Averaging	None
ML Algorithm at Server	None	Anomaly Detector of Anomaly Scores

8.4 Architecture and Components

Table 32 provides the definitions for the terminology and software components found in the STADE architectural diagrams that follow. Figure 45 illustrates the STADE cloud-based architecture. Each site or geographic region has a separate STADE instantiation and processes multi-dimensional data streams needed to calculate the anomaly scores. The diagram illustrates four (4) STADE sites, but there is no technical limit as to the number of sites within an architecture. There is also no technical limit to the geographical location of the sites as communication may occur across the public internet.

To summarize the CONOPS, each site receives relevant data (e.g., Site #1 receives streaming data from source #101), performs anomaly detection on that data using a SAD, and transmits the score to the global score repository for processing by the FAD. This process is repeated upon receipt of each instance of data. The anomaly scores are accumulated from all

SADs and stored in the global repository for aggregation and additional processing, display at a remote DSS, and further processing by the FAD.

Table 32: STADE Terminology and Components

Anomaly Score (AS)	The output of an algorithm that measures the anomalous degree assigned to multivariate data received over some time.
Stream Anomaly Detector (SAD)	A site-specific implementation of an anomaly detection algorithm using online SGD. Model parameters are updated continuously.
Decision Support System (DSS)	A digital, model-driven information system that processes and displays anomaly information and assists with decision-making.
Global Repository	A STADE site that can persistently aggregate, store, transmit, and process anomaly scores from other STADE sites.
Publish and Subscribe Message Bus	A publish-and-subscribe protocol to asynchronously transmit data between STADE sites and the global score repository.
Message Component	Software that manages the interfaces with the message bus and messages between components at a STADE site.
Federated Anomaly Detector (FAD)	An implementation of an anomaly detection algorithm that operates on anomaly scores in the global repository.
STADE Architecture	A group of two or more STADE Sites that communicate via a pub-sub message bus and that includes a global repository.
STADE Site #	A numbered geospatial instantiation of STADE. Each site processes unique data and executes unique anomaly models.
Storage Component	Includes software that manages storage at a STADE site. All data received into a STADE site is stored locally.
Stream#	A numbered data stream for processing by the stream component. Multiple streams sources may enter a STADE site.
Stream Component	Software that processes incoming streams and performs stream quality control.
Workflow Component	Software that orchestrates and manages the sequence and flow of STADE components and a Stream Anomaly Detector (SAD)

In addition to the SAD, each STADE site consists of four components, (a) a workflow component, (b) a stream component, (c) a storage component, and (d) a messaging component. There is also an interface to the publish-and-subscribe message bus used to transmit anomaly scores to the global score repository. These components are notional because an actual instantiation of STADE may combine multiple functions into one component or may use the operating system or commercial cloud computing capabilities (e.g., a message bus) instead of a physical software component.

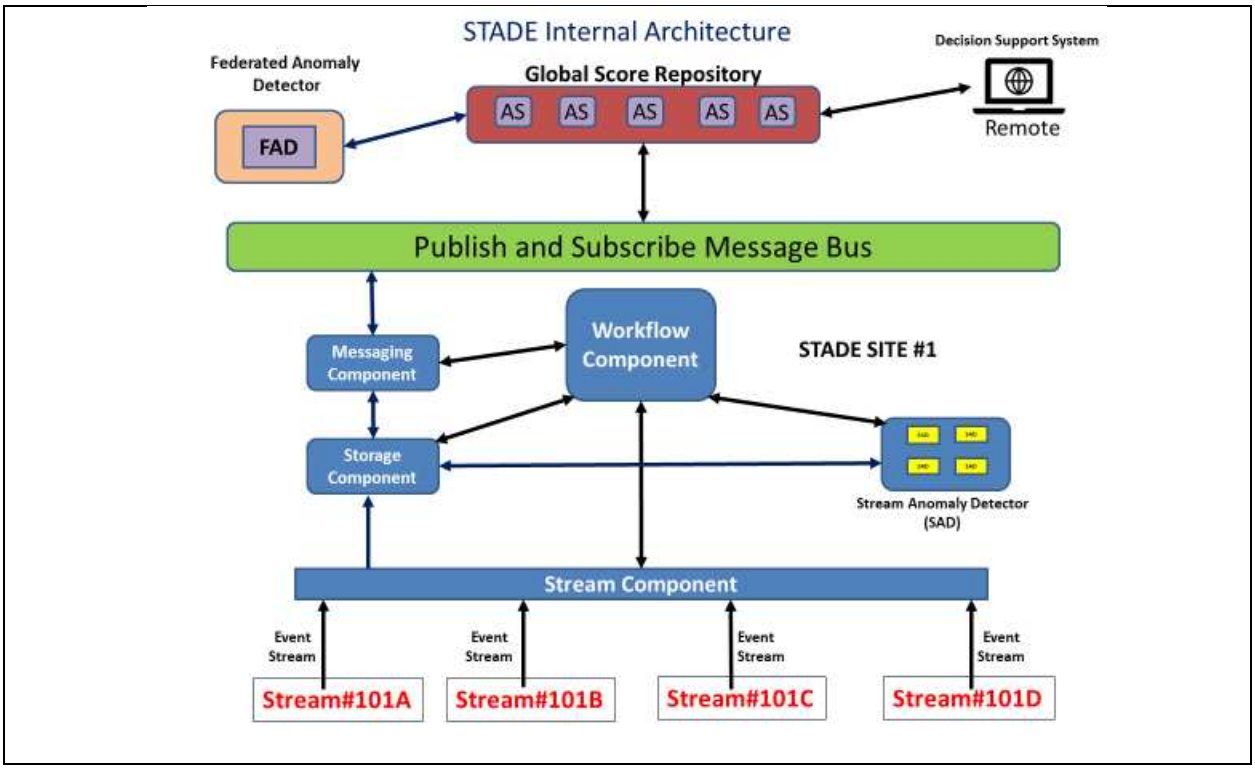


Figure 45: STADE Site Internal Architecture

The workflow component provides the heartbeat of STADE and orchestrates the internal flow of control, such as directing messages to the message processor or controlling the transfer of data. Workflows are always a two-way interaction between the workflow component, the three components (Message, Storage, Stream), and the SAD. Workflows can also be expressed implicitly, or hardwired into the source code for smaller environments without the need for a dedicated workflow engine. Formal workflows are preferable to hardwired workflows because models are easily modifiable and can be placed under configuration management.

The stream component controls the ingestion and processing of externally transmitted streaming data. Upon receipt, the stream component forwards the packets to the storage component. The storage component persists the data into storage and forwards a copy to the SAD, which then executes the anomaly detection algorithm on the newly received data. The SAD, in turn, calculates the anomaly score and submits the results back to the storage

component. The storage component forwards the anomaly scores to the message component, which, in turn, inserts the newly calculated anomaly scores onto the publish-and-subscribe message bus for transmission to the global score repository. Although STADE is a relatively simple system, the workflow becomes surprisingly complex, with just a single data flow and four software components.

The glue that supports communication with the global score repository is the publish-and-subscribe message bus. Scalability is also enhanced by the publish-and-subscribe middleware to promote loose coupling. Publish-and-subscribe is a messaging pattern that provides for increased network scalability with loose coupling, highly desirable design characteristics of distributed systems. Scalability is increased because STADE sites can filter messages if required, communicate asynchronously, and are not tightly coupled, as would be the case in a client-server architecture. By providing a dynamic network topology, publish-and-subscribe simplifies implementation, as a new site participating in a STADE architecture has no impact on the existing set of sites that are currently in operation.

Since the product of STADE is an anomaly designation, a critical component is SAD. Alternative SADs based on different TML and DDN algorithms may be inserted without modification to the other architectural components. If a higher-performing SAD is discovered, that detector can be inserted into the STADE instantiated architecture.

8.5 Federated Anomaly Detector (FAD) and the Global Score Repository

The FAD is an anomaly detection algorithm that operates on the anomaly scores stored in the global repository. At first glance, the rationale for performing anomaly detection on the set of anomaly scores is not entirely apparent. However, the approach here follows meta-analysis, a statistical procedure used in many different scientific disciplines for combining data in multiple

studies. In this case, the score repository is combining anomaly scores from multiple sites or regions. The meta-detector approach attempts to identify collective anomalies by considering time-stamped anomaly scores from one site or region only within the context of the anomaly scores at other sites and regions.

Consider three (3) geographically distinct STADE sites. Anomaly scores are submitted to the score repository simultaneously by these three sites. Anomaly scores will vary region by region for a variety of reasons, but the anomaly scores may be related depending on the domain characteristics. For example, a significant increase or decrease in the anomaly score at the same time at all sites might indicate a non-anomalous occurrence (e.g., a benign solar disturbance that has a temporary effect on sensor recordings). When considering point anomalies only, each site would be designated anomalous. However, collectively, there is no anomaly; only when an increase or decrease occurs in one site vis-a-vis the other two sites is there an anomaly. These types of relationships are difficult to decipher if there are more than a few sites, which is why the meta-detector approach through the FAD is incorporated into STADE. Which anomaly detection algorithms are appropriate for the FAD? Within STADE, FAD algorithms are pluggable as there is no prior reason to prefer one anomaly detection algorithm over another.

8.6 Algorithms and Estimation

In most neural network-based commercial applications, training occurs offline, often with specialty hardware, and the resulting model deployed to production in the cloud or to consumer devices. For example, in a smartphone language translation app, training occurs offline using supervised learning techniques with millions of text-to-speech or text-to-text samples, and the resultant highly performant binary is deployed to the smartphone with a minimalistic footprint. The text-to-speech model is static and does not change over time.

STADE cannot use this approach. First, STADE supports unsupervised learning; there are no training datasets. Second, with streaming spatiotemporal data, the ability to capture concept drift to the extent possible is critical to system performance. Recognition of concept drift is particularly crucial with mission-critical applications (e.g., military combat systems) where adversaries intentionally change behaviors in order to deceive. Third, STADE is designed to leverage the information provided by other STADE sites through the FAD; this exchange of information would not be possible if models were deployed statically.

In many domains, streaming data may arrive faster than the ability of the DNN to execute SGD in real-time. Deployment of static, pre-trained models is possible, but the parameters would become out-of-date quickly in the presence of concept drift. There are three approaches to address the issue of online training of neural networks with streaming data. The first approach is an optimizer that implements online stochastic gradient descent (SGD) training with backpropagation, the second approach is delayed training with batch SGD, and the third approach is a variant of FL. This third approach, FL, is not considered further because of the differences in the underlying STADE requirements, as noted in Table 31.

To summarize, in highly dynamic environments, model parameters may change over time, often subtlety, a problem known as concept drift. Therefore, learning algorithms should adapt to the changing parameters; the learning needs to be online and in real-time. The learning algorithm also needs to be adapted for the velocity of the data. If, for example, the receipt of streaming data is faster than the processing of that data, then the algorithm needs to be modified to support the velocity of the data stream.

8.6.1 STADE SGD ALGORITHMS

With the first approach to processing streaming data, online SGD, training occurs one sample at a time upon arrival at the processing site. So, with a persistently running SGD implementation, at time t-1, the arriving sample could be used to update the SGD parameters. At time t, the DNN model estimated using the parameters of the t-1 sample could be used for anomaly scoring and also to update the DNN parameters using SGD again. This process of anomaly scoring and parameter update would continue upon receipt of each new sample from the data stream.

Online SGD is significantly faster than batch gradient descent and could be adapted for use with streaming data. However, the SGD technique generally exhibits higher variance compared to batch SGD and can cause significant fluctuations in the parameter estimates. With higher variance, recognition of concept drift and other data imperfections is less precise than using the traditional batch SGD with DNNs.

The second approach is the delayed parameter estimation using batch SGD. Under this approach, streaming data is accumulated over time by the STADE storage component and used in the SAD training algorithm in the same way as offline SGD estimation. The newly estimated parameters are then applied to the newly received streaming data in an anomaly scoring algorithm. The SAD training algorithm is restarted once again, including the most recently received streaming data. Under this approach, the model parameters are out-of-date only to the extent of the time required to retrain the model with newly received streaming data.

The training time under SGD may exceed the mean period of receipt of new streaming data. In these circumstances, the second approach, delayed training with batch SGD, may be combined with SGD to create a hybrid algorithm. Under this hybrid approach, an optimizer with

SGD with backpropagation is persistently executing and processing the latest set of streamed data. Upon completion, the SGD is restated, and execution is started again with the latest set of streamed data.

The goal of gradient descent is to minimize the objective function $J(\theta)$, where parameters $\theta \in \mathbb{R}^n$, the set of real numbers in n dimensional space. With gradient descent, parameters are updated in the opposite direction of the gradient of the objective function with respect to the parameters θ, given by $\nabla_\theta J(\theta)$. The learning rate ŋ determines the size of the step used in the gradient descent algorithm. Let x_i^t denote the multivariate input streams at location i at time t and may include lagged values to capture the impact of time. Since the model at location i is independent of the model at location j, for brevity, we can omit the location subscript. With unsupervised learning, there is also no targeted sample, so the goal is to minimize the reconstructive error of the input sample. Let $x^{t\prime}$ be the target value at time t. Therefore, the update equation for stochastic gradient descent is given by (8.1) equation:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^t; x^{t\prime}) \tag{8.1}$$

Therefore, SGD with backpropagation performs one update at a time.

Note that in most batch SGD approaches, the input data is shuffled, and samples are randomly drawn before running the backpropagation algorithm so that the most recent streamed data may not be the sampled as part of the parameter update cycle. Processing the most recently received stream data would be preferred over shuffling if there was a desire to identify concept drift. The reason for the random draw is that gradient descent is a multi-pass algorithm. The number of epochs is a hyperparameter that specifies the number of complete passes, both forward and backward, through the training data. In order for the parameters to be learned,

different samples must be used through each epoch. There are usually many epochs and many iterations through the samples.

Most SGD software packages support options for no shuffling and only one (1) epoch per time cycle. Note that the SGD algorithm overshoots the global minimum during the early stages of training. Therefore, the recommendation in the literature is to slowly decrease the learning rate as training proceeds to mitigate the large parameter swings caused by the use of only a single sample in SGD. The above discussion suggests that parameter estimation using SGD is challenging, albeit less problematic than attempting to use a recurrent neural network (RNN) architecture in an online, streaming environment. Online SGD challenges include general conversion to a global minimum, selection of the proper learning rate, selection of the approach for including of time-lagged features, and the approach to capturing concept drift over time.

Figure 46 below displays the STADE estimation algorithm #7 for SGD. This algorithm is appropriate for autoencoder and other feedforward architectures but not appropriate for recurrent neural networks.

Figure 47 below displays the STADE algorithm #8 for the Delayed Parameter Estimation algorithm. With delayed parameter estimation, traditional batch SGD is used. Gradient descent is restarted with the newly received data after the old gradient descent algorithm completes. So, the trade-off is better parameter estimation from the use of batches at the expense of some stale parameters existing for short periods. Algorithm #7 and algorithm #8 are similar in most aspects. For example, algorithm #8 is identical to algorithm #7 under the assumption that SGD executes faster than the receipt of the stream (so that there is no delayed execution) and that the batch size =1 without shuffling.

Algorithm 7: Perpetual Stochastic Gradient Descent (SGD)	
INPUT:	incoming streaming sample: x_i^t, # of epochs: E==1, learning rate: ŋ accumulated dataset: X
OUTPUT:	parameters
STEPS:	*parameters* ← train a feedforward network with SGD append x_i^t to X loopOverEachEpoch E sample = get_data(X,1) params-grad = evaluate-gradient(loss-function, sample, parameters) parameters = parameters – (learning rate ŋ * params-grad)

Figure 46: Stochastic Gradient Descent

Algorithm 8: Delayed Parameter Estimation (with Batch Gradient Descent)	
INPUT:	Incoming Streaming sample: x_i^t, # of epochs: E, Learning Rate: ŋ Batch size: n Accumulated Dataset: X
OUTPUT:	parameters
STEPS:	*parameters*← train a FeedForward Network with gradient descent append x_i^t to X if(Delayed Parameter Estimation is not executing) loopOverEachEpoch E random-shuffle(X) loopOverEachMiniBatch n get_data(X,n) params-grad=evaluate-gradient(loss-function,sample,parameters) parameters = parameters – (learning rate ŋ * params-grad)

Figure 47: Delayed Parameter Estimation

8.7 STADE Instantiation

A given STADE instantiation consists of an operating environment, infrastructure software, SAD and FAD, decision support system, and a workflow. Each of these elements is discussed below.

8.7.1 Operational Environment

The execution of the case studies on operational equipment was preferable to a laboratory environment. The belief is that the most information on the suitability of STADE architecture could be garnered by using cloud-based, globally distributed equipment that communicated over

the public internet. The Microsoft™ Azure commercial cloud was chosen as the platform to conduct the case studies. Azure has data centers located throughout the world and provides several enterprise-level services that could potentially be mapped to components required for an instantiation of STADE.

Six (6) Azure datacenters were chosen to host a STADE instantiation. These data centers were located one in each continent, Africa (South Africa), Asia (India), Australia, Europe (United Kingdom), North America (USA), and South America (Brazil). Figure 48 provides a map of the physical locations of the datacenters hosting STADE. One (1) location is also designated as the global score repository hosting the FAD. The North America STADE implementation is designated as this global score repository.

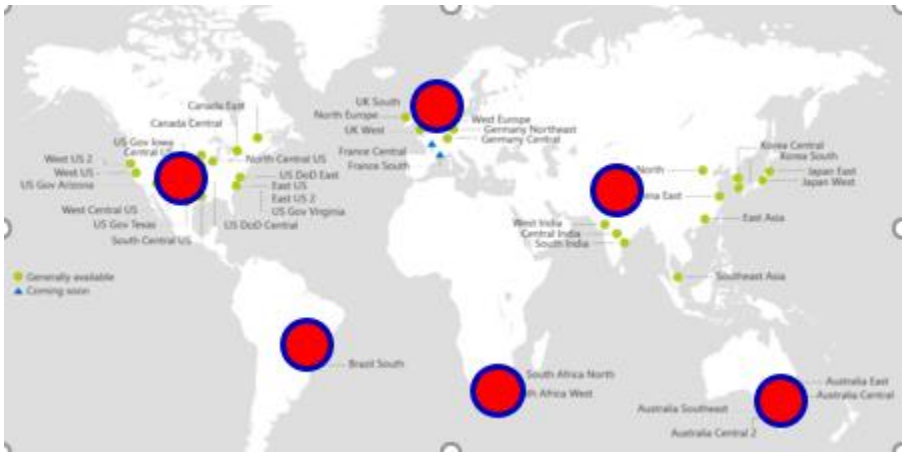


Figure 48: STADE Case Study Testbed

Each Microsoft™ Azure datacenter hosted one or more STADE virtual machines. No additional Azure software or services were used, and all cloud services were accessed remotely via the remote desktop connection protocol.

8.7.2 Infrastructure Software

Table 33 relates the components of the STADE specification to the components used in the case studies. The components include services provided by the Microsoft™ Azure cloud, interfaces to cloud-based database systems, and custom software programs that execute on local computers or virtual machines in the cloud, The most critical components include COSMOS DB, which serves as a cloud database and provides capabilities for asynchronous communications similar to the publish-and-subscribe message broker, and Azure Functions, which supports workflow capabilities. The DSS is based on a web-based interface; all end-user interactions are completed through interactions with a global map.

Table 33: STADE Case Study Software Components

Stream Anomaly Detector (SAD)	Shallow/Deep Autoencoder / HBOS
Decision Support System (DSS)	Azure Maps Web-Based Map
Global Repository	Cosmos DB
Publish and Subscribe Message Bus	Microsoft™ Azure Functions
Message Component	Microsoft™ SignalR Azure Service
Federated Anomaly Detector (FAD)	Shallow/Deep Autoencoder / HBOS
Storage Component	Microsoft™ Azure COSMOS DB
Stream Component	Custom Software
Workflow Component	Microsoft™ Azure Functions Custom Software

8.7.3 SAD and FAD Anomaly Detectors

There are two types of anomaly detectors in the STADE architecture, a SAD executing at multiple local sties and a FAD executing at a single global site. Both the SAD and FAD are configured to ingest inputs as they arrive, perform online learning, and calculate the anomaly score. The input to the SAD is high-dimensional (e.g., sensor data), while the input to the FAD is anomaly scores generated from the SAD sites. These anomaly scores are then transmitted via the publish-and-subscribe messaging infrastructure for storage at the global repository.

The TML and DNN algorithms previously discussed are all candidates for employment as the SAD and FAD. For example, the DNN shallow and deep autoencoder (SDA) algorithm might be designated as the SAD, and the TML Histogram-Based Outlier Score (HBOS) algorithm was designated as the FAD. These SDA and HBOS algorithms might be selected because both were reasonably effective in identifying anomalies in the experimentation. Other combinations of algorithms are a reasonable combination. The SDA is estimated using perpetual stochastic gradient descent (Algorithm 7). Different SADs could have been selected for different sites as the STADE architecture is pluggable and loosely coupled; however, deployment of STADE is simplified if the same set of SADs is installed throughout the architecture.

8.7.4 Decision Support System (DSS)

The DSS is provided as a web site that communicates with the global repository through the publish-and-subscribe message bus and displays streaming point data and anomaly scores reported from each STADE site. While the architecture diagrams display the decision support web connecting to the global repository only, the decision support web can also connect to the individual STADE sites to monitor incoming feeds and view anomaly scores for that location. Note that in the case studies, no analytical algorithms execute on the decision support site, but, in practice, background algorithms would further analyze the scores and provide additional assessments and graphical displays to the decision-maker.

8.7.5 Workflow

Figure 49 displays the sequence of events, or workflow, for the arrival of one sample (e.g., sensor reading) at a given site.

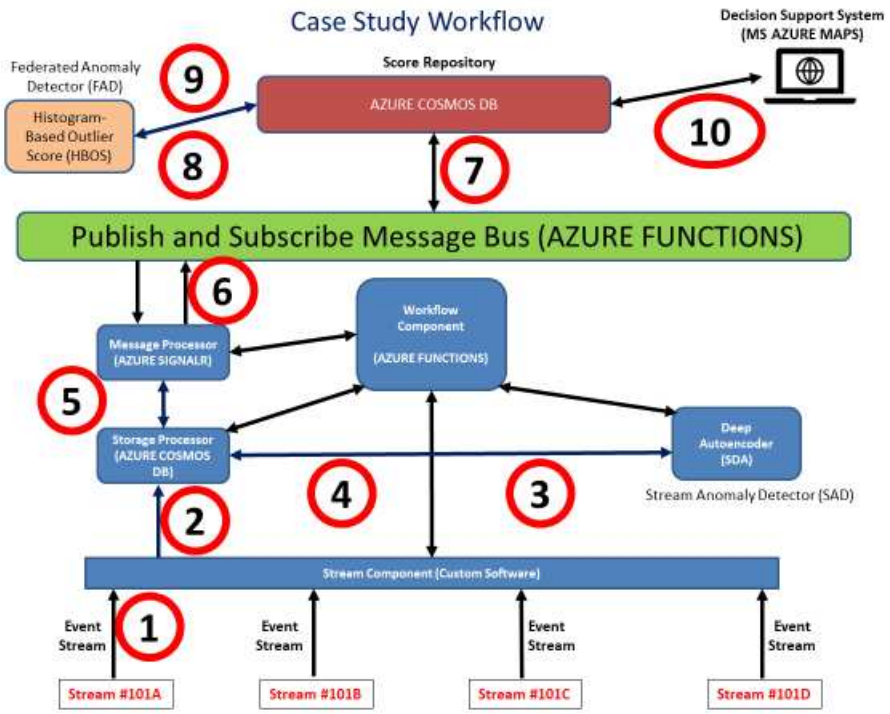


Figure 49: STADE Case Study Workflow

Each number represents one step in the end-to-end chain from receipt to the data to display on the decision support web. While there are ten (10) numbered steps, steps one to six (1-6) are implemented on one virtual machine at one site, while steps seven to ten (7-10) are implemented on another virtual machine at another site. The global repository and the FAD are implemented at the STADE site designated in North America. The most significant potential impact on performance is the message bus, which transmits data over the internet and would be a choke point in the presence of network outages or global latencies.

8.8 Case Study Objectives

Chapters 9-11 describe three (3) case studies that utilize the STADE instantiation described. Chapter 9 describes the global air traffic case study, Chapter 10 describes the earth sciences case study, focusing on earthquakes, while Chapter 11 describes the social networking streams case study. The prime objective of these case studies is to determine if STADE is a

viable architecture for the detection of anomalies in highly dimensional streaming spatiotemporal data. These case studies are proof-of-concept. While STADE is domain-independent, an instantiation of STADE is tailored to the particular problem statement of each case study. STADE can support automated decision making or augment the human decision-making process. Because STADE supports unsupervised learning, the algorithms could potentially identify novel or never-seen (zero-day) anomalies. STADE can be applied either in a local area network clustered or a geographically distributed cloud (e.g., global sensor network). The instantiation of STADE described above is cloud-based. The algorithms exploit the information contained not only in the values at a point in space-time, but also the sequences of data, or spatiotemporal anomaly detection.

8.9 Summary

STADE is designed to address spatiotemporal anomaly detection in a distributed, cloud-centric, domain-independent application domain. Anomaly detectors execute at local STADE sites and report the anomaly score to a centralized Federated Anomaly Detector (FAD) site for further analysis or display on the remote decision support system. The centralized repository provides further analysis of the reported anomaly scores. With multiple geospatially distributed sources of information, the decision-makers can make informed decisions regarding the existence or non-existence of a spatiotemporal anomaly.

The description of STADE in this chapter included a mapping of the architecture to specific products and technologies. This instantiation included several existing open source technologies and commercial products since programming STADE from scratch is cost-prohibitive. Each use case has unique requirements and may require a unique instantiation of the STADE components

The design goal of STADE is to process streaming spatiotemporal data for anomaly detection. Most anomaly detectors have long training times that make online neural network learning with streaming data difficult. Two practical solutions to this problem are the use of stochastic gradient descent and a delayed parameter estimation approach. Spatial considerations are addressed within STADE by running separate models per region; there is no cross-region pooling of data. Temporal issues are addressed by using an anomaly detection algorithm that has built-in multivariate support for time or sequences (e.g., ED-1D-CNN) or by direct inclusion of time as part of the feature set. For example, lagged response variables from a time series can be introduced as features in autoencoders without the need for explicitly using a sequentially dynamic model (e.g., RNN).

8.10 Related Work

Studies of streaming anomaly detection utilize various time-series techniques with univariate data. The Numenta Anomaly Benchmark (NAB) [181] and [182] includes a set of univariate time-stamped datasets with anomalies annotated by a well-defined human labeling process. Specially designed datasets were created to allow the comparison and scoring of various algorithms that support streaming anomaly detection. Anomaly detectors assign scores on the test data based on the parameters estimated from the training data. The NAB scoring system, in turn, calculates an overall performance metric based on a set of rules. For example, the performance metric penalizes detectors that trigger higher false alarms than expected. NAB is designed to allow a comparison of alternative univariate techniques against a single baseline.

TML techniques designed for real-time streaming anomaly detection include hierarchical temporal memory (HTM) [183] and [184], random cut forests [185], and seasonal trend decomposition based on loess (STL) combined with seasonal autoregressive integrated moving

average (SARIMA) [186]. Other anomaly detection applications designed specifically for processing online streaming data include domain-independent processing of complex event streams through 'STREAM-LEARNER' [187], wireless sensor networks [188], road traffic conditions [189], unmanned aerial vehicles (UAVs) [190], and network intrusion detection [191]. Choudhary et al. [192] present an analysis of the runtime trade-off of various techniques to process real-time streaming anomaly detection.

STADE is designed to support distributed real-time anomaly detection in multiple domains, including sensor networks. Ball et al. [193] provide a survey of representational learning techniques for remote sensing and sensor networks. Budalakoti et al. [194] develop an anomaly detection system called 'sequenceMiner' that detects anomalies by recording and analyzing the symbol sequences of switch sensors in the cockpits of commercial airlines. Hayes and Capretz [195] provide a contextual anomaly detection model for sensor data. Mohammaddi et al. [196] provide a survey of neural network applications for big data and streaming analytics. Muallem et al. [197] provide a survey of hoeffding tree algorithms, a TML technique, for streaming cyber anomaly detection applications. Xie and Chen [198] address anomaly detection with the elimination of data redundancy in sensor data streams. Other studies of sensor anomaly detection include [199], and streaming data include [200].

FL inspires the STADE architecture by the 'attention' mechanism [92] that is popular today in the DNN language translation literature. The attention mechanism is an evolution from an earlier architecture based on the encoder-decoder recurrent neural network model (ED-RNN). Models that use a combination ED-RNN with or without attention include [201], [152], [22], [202], [203] and [204].

The STADE architecture is based on FL and incorporates the concept of a centralized repository to maintain and exchange anomaly scores across geographically dispersed clients. Within this architecture, all algorithms are trained locally, and the results coordinated across regions. There have been numerous projects by Google™ and others to formulate distributed representational learning algorithms primarily to support image processing tasks. Distributed processing is typically within a data center or in a computer cluster and not geographically dispersed as proposed here. Dean et al. [205] describe a system called DistBelief consisting of thousands of CPU cores in data centers used to train large models with sixteen (16) million images. Note that while asynchronous stochastic gradient descent procedures can be parallelized in image processing convolutional neural networks, parallelization is not possible with a recurrent neural network or autoencoder architecture.

Moreover, STADE architecture is designed to support disadvantaged locations at the network edge. For a comprehensive survey of various approaches to large-scale distributed training of DNNs and various formulation of the stochastic gradient descent algorithm, see Chahal et al. [206]. Williams and Zipser [207] propose an algorithm for continually running RNNs; however, the performance is worse than the performance of a traditional RNN and is not suitable for high tempo streaming applications.

In summary, research on real-time streaming anomaly detection has been spotty and has focused on univariate problem domains and non-scalable machine learning techniques. There does exist online streaming stochastic gradient descent algorithms, but their performance is still lacking and not suitable for all streaming applications. New, more agile algorithms are needed to process streaming data for high-tempo, real-time anomaly detection.

CHAPTER 9 – STADE CASE STUDY #1: GLOBAL AIR TRAFFIC (GAT)

9.1 GAT Background

The purpose of the GAT case study is to identify, in near real-time, anomalies in global air traffic. The parameters would be learned from past non-anomalous flights and applied to ongoing flights. A decision-support system would then provide alerts to air traffic controllers (ATC), commercial airlines, and aircraft manufacturers of possible anomalies that require attention.

One obvious question is, why is a DNN or another set of sophisticated techniques are required in this scenario? Why not analyze the raw data? The reasons are twofold. First, the amount of raw data would be overwhelming for a decision-analyst or a management information system. Second, the data relationships are complex, and a traditional management or database system would unlikely to uncover the complex dependencies and attribute interrelationships necessary to identify a multidimensional anomaly.

9.2 GAT Architecture Design Decisions

The GAT case study architecture follows closely the STADE architecture described in Chapter 8. Note that, in this particular case study, because data feeds are through the public internet, the geographic distribution of the processing nodes is not critical. Moreover, the publish-and-subscribe infrastructure that is part of the STADE global cloud essentially abstracts away the concept of geo-location. Nevertheless, for proof-of-concept and demonstration purposes, STADE processes nodes have been distributed to the various Microsoft Azure cloud sites, as described in Section 8.7.1.

The GAT case study is designed to support global operations involving thousands of simultaneous flights. Given the limited available data, Streaming Anomaly Detector (SAD)

models can be partitioned either by country-of-origin, commercial carrier, or aircraft callsign. The preferred partition would be the airport origin and airport destination pairs because anomalies happen within a particular route. This data is inferred based on trajectories of the takeoff and approach. For demonstration, country-of-origin will be used to partition the SAD models; each country-of-origin is a separate SAD model. The centralized FAD would collect the anomaly scores on a per-country basis. For example, if flights originate from twenty (20) different countries, there would be twenty different FAD scores per period. The FAD collects the highest SAD score every five (5) minutes from each origin country unique model, but the FAD time interval parameter can easily be changed to fit the architecture.

Both the Streaming Anomaly Detector (SAD) and Federated Anomaly Detector (FAD) technique selected for this case study are the Encoding-Decoding Recurrent Neural Network (ED-RNN). Parameters are estimated using perpetual stochastic gradient descent (Algorithm 7) described in Section 8.6.1.

9.2.1 GAT Anomaly Definition

There are limitless examples of potential anomalies, such as the location (latitude and longitude) along an aircraft route, the altitude at that location along a route, and airspeed for a particular aircraft. Anomalies are domain-dependent, however. For example. [208] notes that an anomaly might be a large commercial airliner overflying a terrain in the vicinity of a large international airport.

Since the anomaly techniques are multidimensional, all relevant flight attributes enter into the anomaly scoring algorithm so that a specific anomaly definition is not required. The flight attributes are discussed below. Stream Anomaly Detection (SAD) scores are ranked with

the highest ten (10) scored over 24 hours listed below for all flights originating from the United States. The corresponding Federated Anomaly detection (FAD) scores are also listed.

9.2.2 GAT Data Sources

This case study uses data produced by the OpenSky Network (https://opensky-network.org/). Most commercial aircraft have MODE-S hardware transmitters that emit ADS-B messages. The OpenSky Network is a community-based, crowd-sourced collaborative effort to collect air traffic data from more than one-thousand Mode S sensors located throughout the world [209]. These sensors collect the ADS-B generate messages that track the state of an aircraft at a point in time, including position, velocity, and aircraft identity. Table 34 describes the data provided through the OpenSky Network and delivered in real-time via the internet.

Table 34: Global Air Traffic Data Elements

Data Element	**Description**
icao24	ICAO24 address of the transmitter in a hex string representation
Callsign	Callsign of the aircraft
Origin Country	The origin-country of the flight
Last Time	Time since the last position report
Last Contact	Seconds since last received message from this transponder
Longitude	In ellipsoidal coordinates (WGS-84) and degrees.
Latitude	In ellipsoidal coordinates (WGS-84) and degrees.
Altitude	Geometric altitude in meters
Velocity	Speed over ground in meters per second
On the Ground	True if the aircraft is on the ground
Heading	In decimal degrees, where 0 is north
Vertical Rate	In meters/second. The incline is positive, and decline is negative
Sensors	The serial number of sensors which received messages from this aircraft
Barometric	Barometric altitude in meters
Squawk	Transponder code, aka Squawk.
SPI	Special Purpose Indicator
Position Source	Origin of the position: 0 = ADS-B, 1 = ASTERIX, 2 = MLAT, 3=FLARM
Type code	The aircraft model type
Origin	Origin Airport
Destination	Destination Airport

One additional issue relates to the use of location attributes in the SAD and FAD. For the GAT case study, the exact latitude and longitude are relatively unimportant, and only a relatively

small geographic bounding box has relevance. With a simple formula, the latitude and longitude are converted into a unique integer for entry into the models. The simple formula is: (latitude +90)*180 + longitude. This formula is equivalent to a hashing algorithm in computer science. With this formula, every point on the map can be converted into an integer that designates a unique bounding box accurate to one (1) degree.

9.3 GAT Case Study Results

Figure 50 below displays a snapshot of the GAT map of United States air traffic at a given point in time by exact latitude and longitude (not the integer conversion). This flight display is web-based and animated; flight locations on the map are updated upon receipt of the flight reports (i.e., samples), and plane icons are clickable. Flight reports (i.e., samples) are received every 5 to 10 seconds following the architecture and workflow described in Chapter 8.

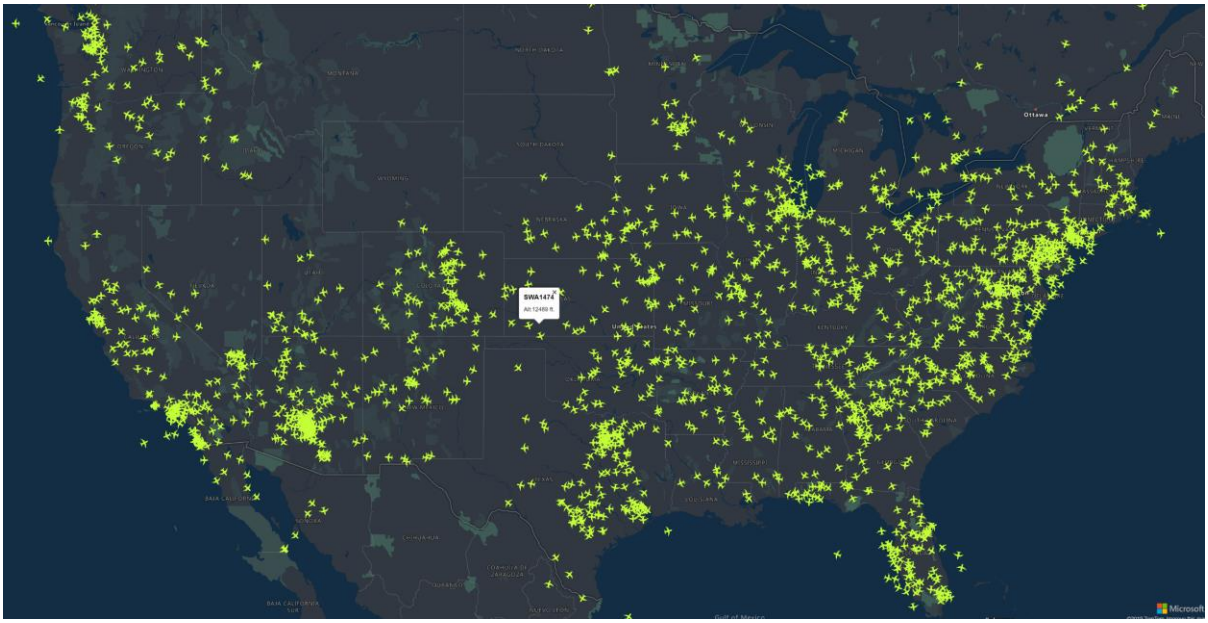


Figure 50: Global Air Traffic Decision Support Map

Table 35: Global Air Traffic Top 10 Anomaly Report (Over 24-Hour Period)

	Date	Call Sign	SAD	FAD	Latitude	Longitude	Velocity	Heading	Vertical	Altitude
1	2020-03-02	OMA154	188.8	Yes	47.14	11.45	250	113.1	6.5	10.7
2	2020-03-02	AZA22FL	13.9	No	50.51	0.47	236	144.5	6.5	9997
3	2020-03-02	TCRSD	13.9	No	59.20	-8.65	221	319.4	0.3	12687
4	2020-03-02	AAY96	12.8	No	37.64	-120.71	147	295.9	10.4	2514
5	2020-03-02	N417EP	12.8	No	28.91	-80.92	60	219.11	0.0	1028
6	2020-03-02	JME508N	12.7	No	32.29	-108.22	297	105.7	0.3	12694
7	2020-03-02	VTI973	11.1	No	19.01	73.92	200	156.7	22.1	3787
8	2020-03-02	UAE542	11.1	No	25.29	56.59	271	100.3	4.87	8107
9	2020-03-02	UAL555	10.9	No	32.73	-116.94	101	109.1	6.17	1226
10	2020-03-02	KAL646	10.8	No	25.73	122.17	296	35.3	0.0	11811

STADE was able to identify hundreds of flight anomalies in real-time in thousands of flights over 24 hours. The model was trained with historical data from the previous thirty (30) days. The FAD found only one 'federated' anomaly for the corresponding time-stamp. This anomaly may be a result of a data error since the reported altitude was ten (10) meters. This result may be an outcome of the artificial partition of the model by country-of-origin. An anomaly in the location of a flight in the United States is unlikely to be related to an anomaly in Great Britain. As previously noted, an operational system would likely partition the models by source-destination airport pairs. The exact cause of these anomalies would require further analysis, perhaps by air traffic controllers. Note that without the inclusion of an automated explainable AI module, a subject on ongoing research in the ML community, the explanation of the anomaly score is complicated.

9.4 GAT Related Work

Few published studies have applied DNN anomaly detection techniques to the air traffic domain. There have been statistical applications of anomaly detection to air traffic issues. Tanner and Strohmeier [210] utilize the OpenSky network to detect anomalies in air traffic patterns and runway use. Other studies use visual satellite data combined with the Opensky network to build a flight anomaly detection dataset [208].

CHAPTER 10 – STADE CASE STUDY #2: EARTH SCIENCE (ES)

10.1 ES Background

The objective of this case study is to investigate earthquake data as provided by the U.S Geological Survey (USGS). The goal is to identify anomalies in sequences of earth sensor readings that may portend a future earthquake or aftershock. The goal here is not to forecast earthquakes, which is a supervised learning problem but to identify anomalies in data sequences that may support explanations of near-term future geophysical events.

Under the Earthquake Hazards Program, the USGS uses statistical modeling techniques to make probabilistic predictions such as: "the chance of an earthquake of magnitude three (3) or higher is > 99%, and it is most likely that as few as 42 or as many as 230 such earthquakes may occur in the case that the sequence is reinvigorated by a larger aftershock." Multivariate anomaly detection techniques could enhance these types of predictions.

10.2 ES Architecture Design Decisions

10.2.1 ES Anomaly Definition

Similar to the GAT case study, anomalies are based on SAD scores. Each network is a separate SAD model. There are approximately fifteen (15) networks throughout the world that collect earthquake data. Each network provides its anomaly scores to the FAD. As before, the SAD and FAD technique selected for this case study is the Encoding-Decoding Recurrent Neural Network (ED-RNN).

10.2.2 ES Data Sources and Design

The earthquake data is provided by the U.S. Geological Survey using the GeoJSON Javascript Object Notation (JSON). GeoJSON is similar to Extensible Markup Language (XML) but is less formal. GeoJSON provides a standard approach for defining geospatial information

such as geometry, attributes, bounding boxes, and projection information. GeoJSON is a standard published by the Internet Engineering Task Force (IETF). RFC 7946 was published in August 2016 and is the standard specification for the GeoJSON format. GeoJSON earthquake data updated every sixty (60) seconds.

Table 36 displays the minimalist data elements that ARE used in the analysis. There are many other data elements, but most are related to measurements of the quality of the data inputs. Included is a quality measure, RMS, which measures the fit of the observed arrival times to the predicted arrival times for the event location. The higher the RMS, the greater the uncertainty of the data associated with the event.

Like the GAT case study, latitude and longitude are converted into a more straightforward integer representation. Seismologists have a far more sophisticated approach to designate homogenous earthquake faults, regions, and zones, but those definitions require extensive domain knowledge and are outside the scope of this study. Note also that earthquakes of all magnitudes are included in the analysis

Table 36: Earth Science Data Elements

Data Element	**Description**
Time	date and time of the event, also known as the origin time
Latitude	Decimal degrees latitude. Negative values for southern latitudes.
Longitude	Decimal degrees longitude. Negative values for western longitudes.
Depth	Depth of the event in kilometers.
Magnitude	The magnitude of the event. Ranges from -1.0 to 10.0
RMS	The root-mean-square (RMS) travel time residual, in seconds. Used to measure the quality of the data.
Net	The reporting network for the event. Network values include ak, at, ci, hv, ld, mb, nc, nm, nn, pr, pt, s e, us, uu, uw
ID	The unique identifier for the event
Place	Textual description of named geographic region near to the event.

10.3 ES Case Study Results

Figure 51 displays all earthquakes of any magnitude throughout thirty (30) days by exact latitude and longitude. The darker the pink color, the higher the number of earthquakes. Note that the display traces well-documented fault lines and areas of high earthquake activity (e.g., California).

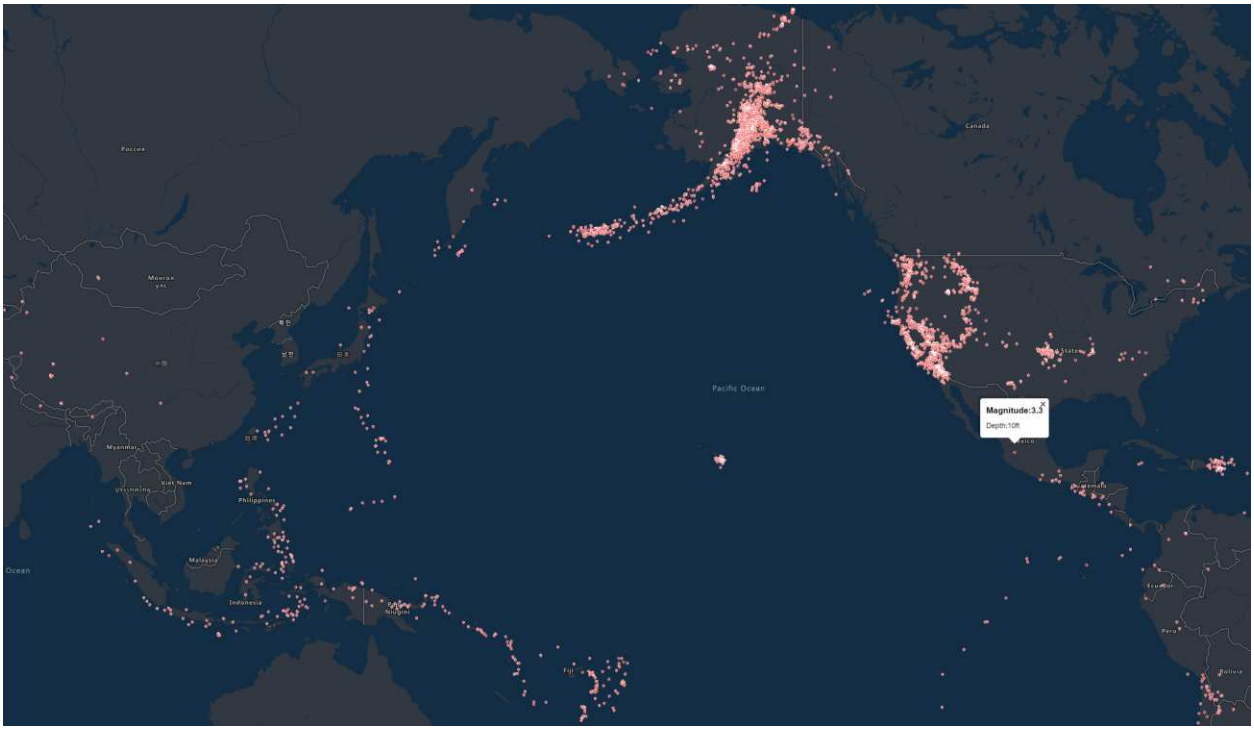


Figure 51: Earth Science Decision Support Map

Table 37 presents the results of the top ten anomalies over one month. While the reporting is global, most of the anomalies were found in low-magnitude earthquakes at shallow depth in California. Since an autoencoder is used in this case study, the highest reconstructive mean-squared error produced the highest scores. Unlike the GAT case study, the sample sizes

are small since few global earthquakes occur during any given minutes, which may account for the appearance of the FAD anomalies.

Table 37: Earth Science Top 10 Anomaly Report (Earthquakes Last 30 Days)

ID	Description	SAD	FAD	Latitude	Longitude	Magnitude	Depth
2632	17km ESE of Anza, CA	13.66	Yes	33.50	-116.50	1.3	10.09
931	4km N of Redwood Valley, CA	12.10	Yes	39.29	-123.21	1.6	6.71
3209	6km NW of The Geysers, CA	11.65	Yes	38.81	-122.80	1.1	1.20
1179	16km ESE of Anza, CA	11.56	Yes	33.50	-116.51	0.5	10.13
863	9km NNE of Kingfisher, Ok	10.73	Yes	35.94	-97.89	1.4	5.77
619	16km ESE of Anza, CA	10.26	Yes	33.56	-116.50	1.0	10.66
3603	5km NW of the Geysers, CA	10.12	Yes	38.81	-122.79	1.1	3.51
1238	8km NW of Anza, CA	9.95	Yes	33.59	-116.74	0.3	12.64
349	16km W of Searles Valley, CA	9.79	Yes	35.76	-117.57	0.8	6.99
1141	16km ESE of Anza, CA	9.69	Yes	33.50	-116.50	1.2	10.83

10.4 ES Related Work

There has not been extensive published research of DNN related unsupervised anomaly detection techniques applied to earthquake data. A few studies have used basic supervised DNN techniques to forecast earthquake magnitudes [211]. Aster [212] provides a high-level overview of statistical modeling earthquake sequences and aftershocks and emphasizes the importance of recognizing, in real-time, earth sequences that create conditions for large aftershocks. Pavlidou et al. [213] provide a time-series analysis of the impact of temperature on twenty (20) global earthquakes.

CHAPTER 11 – STADE CASE STUDY #3: SOCIAL NETWORKING STREAMS (SNS)

11.1 SNS Background

The Twitter™ global feed was selected as an example of an SNS. While tweets are not of particular interest to the research here, the feeds are freely available, spatially and time distributed, and are an endless source of human sensor data that is convenient to use for the case study of STADE. Twitter™ is known as a microblogging platform that also provides a streaming Application Programming Interface (API) supporting globally accessible real-time, text-based tweets. These tweets can be filtered by several attributes, including content, time, and geospatial origin.

Interesting anomaly detection problems can be analyzed using tweets. For example, through various natural language processing (NLP) techniques, a tweet's anomaly score and associated attributes can be examined and used to determine if the tweet is genuine or produced by a malicious bot. Another aspect of the Twitter feed is called sentiment analysis [214]. Sentiment analysis is the process of computationally identifying and categorizing opinions expressed in the text. Objectivity analysis is the process of computationally identifying where there is bias, opinion, or emotion. The goal of the case study is to provide a framework to capture near real-time anomalies in sentiment and objectivity based on time and geographic location of the tweet.

Sentiment and objectivity analysis is a sub-field of the general area of Natural Language Processing (NLP) that processes written material and digitally extracts attributes such as the polarity, subject matter, and the entity who authored the text. For the case study, the sentiment is expressed in terms of polarity and subjectivity. The polarity of a sentence is a measure of whether the author expresses a positive, neutral, or negative opinion. The polarity metric ranges

from -1 (very negative) to 0 (neutral) to +1 (very positive). In most instances, polarity would be zero, or neutral. Anomalous posts occur when the polarity is close to -1 or +1. Subjectivity measures the degree to which a statement of opinion or a statement of fact and exhibits a value between -1 (subjective or opinion) and +1 (objective or fact).

11.2 SNS Architecture Design Decisions

The SNS architecture is identical to the GAT and ES case studies, as described in sections 9.2 and 10.2, and follows the STADE architecture described in Chapter 8.

11.2.1 SNS Anomaly Definition

An anomaly is defined as a combination sentiment and objectivity score that exceeds the projected value in a particular geographic region. In this case study, the geographic region is defined as the country of origin of the tweet, so that separate SAD models are applied to each country.

11.2.2 SNS Data Sources

The tweet object contains a long list of attributes such as the id of the tweet, the tweet text, the set of attributes of the person creating the tweet, whether the tweet is original or a retweet, the number of followers of the person tweeting, and many other features. In the tweet are a set of coordinates that represent the latitude and longitude of the tweet as reported by the user or client application. Also included is a 'place' attribute that may or may not be present in the data stream. When present, this attribute indicates that the tweet is associated but not necessarily originating from a place. The place might be a city, a historical place, an event, and many other possibilities. The coordinates and the place attributes can determine the geographic location of the tweet.

Table 38 provides the selected data elements to use in the SNS case study.

Table 38: Social Networking Streams Data Elements

Data Element	Description
ID	Unique identification associated with the tweet.
Text	Text of the Tweet
Polarity	Calculated:
Subjectivity	Calculated:
Username	User Screen Name of the author of the tweet
Name	User name of the author of the tweet
Profile Image URL	Profile Image
Location	User location from the account profile.
Confidence	The confidence level associated with the tweet.
Latitude	Calculated from available location and place data.
Longitude	Calculated from available location ad place data.
Place	When present, it indicates that the tweet is associated, but not necessarily originating from) a Place.

One potential limitation of tweets is that they are in multiple languages. The Twitter™ API does include support for language translation (e.g., French to English). However, language translation technology is not perfect. Anomaly Detection algorithms that rely on NLP may perform sub-optimally with multi-language streams such as Twitter™.

11.3 SNS Case Study Results

Figure 52 shows a screen snapshot of the Twitter™ feed taken from the case study after processing by STADE. An open-source NLP python library, text blob, is used to translate a sentence into a polarity and subjectivity metric. For example, with 'pizza' as the search word, the sentence "just discovered the best pizza sitting right under our noses" earned a polarity score of .42 (somewhat positive) and a subjectivity score of .27 (somewhat objective). Another sentence, "I hoped you still enjoyed your pizza to the fullest," earned a polarity score of .5 and a subjectivity score of .7. So, the second sentence is slightly more positive and subjective than the first sentence. Throughout the STADE architecture and workflow, these tweets were allocated to geographic regions by the publish-and-subscribe cloud infrastructure.

Figure 53 displays the geographical distribution of tweets within a fifteen (15) minute period, where the tweet text includes the word 'coronavirus.' This figure illustrates the global composition of tweets and the high volume on a topic of global interest.

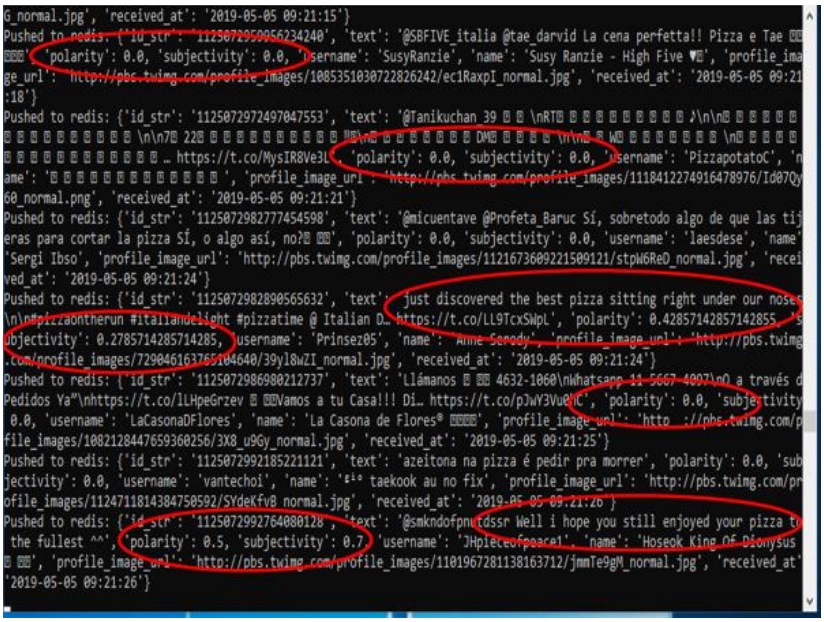


Figure 52: Social Networking Stream (Twitter) Feed

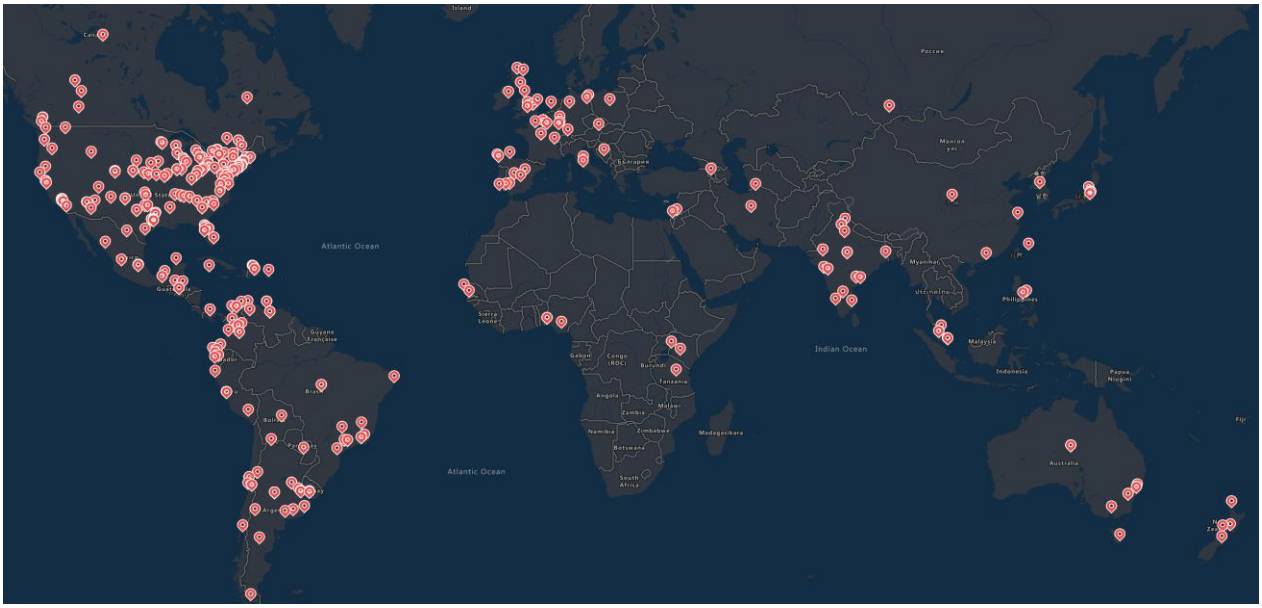


Figure 53: Global 'Coronavirus' Tweets – 15 Minute Period 4/6/2020

Unfortunately, STADE could not identify anomalies in the tweets. First, each tweet anomaly score is autonomous and not dependent on previous tweets, so that tweet sequences are not useful. Second, there was a large number of tweets with values at the extremes. For example, there were many negative tweets (polarity = -1) combined with statements of opinion

(subjectivity = -1). Conversely, there was a large number of very positive tweets (polarity = +1) combined with statements of fact (objectivity = +1). Anomalies must infrequently be occurring. SAD techniques are only useful if there are true anomalies in the data. Similarly, FAD techniques are only useful if there are anomalies in the anomaly scores. If this case study, due to the nature of the subjectivity and polarity text algorithms, within a given country, there lacks variability in the anomaly scores. There exists more variability across countries, but the variability was insufficient to identify anomalies.

The number of incoming tweets was variable depending on the region. Only English language tweets were considered, and the other languages discarded because the sentiment score would be difficult to interpret. The United States experienced a high tempo of incoming tweets, and the online SGD algorithms in the United States site could not maintain pace with the incoming tweets. Such was not the case with the other cloud-based sites, which had a much slower tempo of incoming tweets. The end-to-end time to complete the cycle from tweet receipt through storage at the global repository was roughly five (5) seconds. However, the required time was highly variable depending on the network characteristics, time-of-day, and other factors.

This case study demonstrates the STADE architecture designed to connect spatiotemporal streaming data (e.g., Twitter™ tweets) to a near real-time anomaly detection algorithm. The streaming data is essentially a form of a human sensor. Two distributed algorithms are deployed globally, The SAD and the FAD both running the Encoding-Decoding Recurrent Neural Network (ED-RNN). The design uses a commercial cloud provider, minimalist hardware (low cost, everyday virtual machines), minimalist software (cloud software, open-source software,

small python programs), and the non-dedicated network (public internet) using asynchronous communications (publish-and-subscribe).

The results, while limited by the variability of the subjectivity and polarity metrics, did illustrate that a globally distributed sensor anomaly detection network based on STADE is easy to set up and operate globally. Using a two-step approach, the SAD anomaly detector combined with a FAD, was able to evaluate anomalies in the sentiment and objectivity scores of Twitter™ tweets. Unfortunately, no discernable anomalies were found because there were no rare events. This may also be the reason why the training of the model parameters with online SGD a sample at a time proved to be problematic as the model parameters often failed to learn or change.

11.4 SNS Related Work

The use of social networking applications for anomaly detection experimentation has a practical advantage - the availability of accessible, high volume, streaming data sources freely available for analysis. A social network, in some regards, has similar characteristics to a sensor. Previous applications of neural networks to anomaly detection in social networks include Yu et al. [215], who provides a survey of social media anomaly detection methods. [215] also focuses on the distinction between point anomalies and group anomalies, and distinguishing between activity-based and graph-based methods. Savage et al. [216] provide an overview of online anomaly detection. Tasoulis et al. [217] conduct a statistical approach to sentiment change detection on Twitter™ streaming data, which has similarities to the case study presented here. Castellini et al. [218] use denoising autoencoders to identify fake (or anomalous) followers in twitter streaming data. The use of anomaly detection techniques to identify fake news and data streams has become increasingly important in recent years. Zhang et al. [120] deploy an autoencoder to detect rumors on online social networks.

Different ML algorithms have been applied to this data stream include Naïve Bayes, Max Entropy, and Support Vector Machines (SVMs). Moreover, because of the availability, size, and global distribution, the Twitter™ stream can be used to demonstrate concepts of cloud-based, distributed anomaly detection architectures.

CHAPTER 12 – CONCLUSIONS AND RECOMMENDATIONS

12.1 Conclusions

Unsupervised anomaly detection on high dimensional data is an active research area and touches multiple TML and DNN technologies. Autoencoders, RNNs, CNNs, and GANs deployed singularly and, in combination, have been used to address anomaly detection. A set of TML techniques and six (6) DNN architectures were presented and subject to extensive experimentation using four anomaly datasets. The results indicate that selected TML techniques, such as HBOS, and selected DNN techniques such as autoencoders combined with recurrent and convolutional neural networks performed the best in identifying anomalies. Results are dependent on the quality of the dataset, including the variability of the samples, the accuracy of the anomaly designations, and the degree of imbalance of the training samples. Anomaly detection is a "needle-in-the-haystack" problem domain where a single, universal solution is unlikely to be identified.

One gap in research has been the application of these techniques to streaming spatiotemporal data and the integration with online decision support systems. This research gap is addressed by the Spatiotemporal Anomaly Detection Environment (STADE). STADE is an ensemble approach that combines one or more anomaly detection algorithms with a Federated Anomaly Detector (FAD). The algorithm may be either a TML (e.g., HBOS) or a DNN (e.g., VAE) modified to support stream processing. Borrowing from recent advances in DNN Federated Learning, the FAD is a centralized server that collects and processes anomaly scores from geographically distributed STADE sites. Three case studies, (a) global air traffic, (b) global earthquake measurement, and (c) social media feeds are presented and demonstrate the applicability of STADE to real-world problems. These case studies also demonstrate that the

algorithms perform reasonably well in a highly distributed environment when trained using stochastic gradient descent techniques. The FAD provides valuable anomaly feedback to the individual STADE sites that can be exploited to provide further insights into the spatiotemporal anomaly detection process.

DNNs do have burdensome resource requirements, and processing capabilities may be limited or non-existent at the network edge. STADE is modular, and algorithms can be swapped in and out. Pre-trained neural network models can be successfully deployed at runtime within STADE and re-trained at periodic intervals in a resource-constrained streaming environment.

12.2 Recommendations for Future Work

Research has been hampered by the limited availability of multivariate unsupervised spatiotemporal datasets. Algorithms have been tested using small toy datasets, and researchers have resorted to using synthetic training sets that do not adequately capture domain and distributional diversity. The often-cited KDDCUP intrusion detection dataset used in many studies is twenty (20) years old and has well-documented flaws (Divekar et al., [20]). Investment in the development of a set of broad, high-dimensional benchmarks for steaming anomaly detection is critically needed to advance the state-of-the-art in spatiotemporal algorithms.

A central repository for all neural network-based anomaly detection models and algorithms would be helpful. Researchers often attempt, without success, to reproduce algorithms and implementations other researchers have developed. The experimentation results cited in Chapter 7 did not wholly replicate the findings of other researchers, although the results were close. The performance of these algorithms is sensitive to micro-decisions regarding neural network hyperparameters, optimization assumptions, software packages, and frameworks.

Replication and adaptation of the results of other research are complicated by the complex array of architectural decisions that need to be made up-front.

Alternatives to stochastic gradient descent (SGD) algorithms have not progressed in concert with other DNN technologies and remains an active area of research. The well-cited Real-Time Recurrent Learning (RTRL) algorithm [207] was formulated in 1989 is not suitable for high-velocity anomaly detection problem domains described in the introduction. Long short-term memory (LSTM) and backpropagation through time (BPTT) [219], the standard approach to the estimation of RNNs, is compute-intensive, has unstable estimation properties, and is not suitable to near real-time or streaming parameter estimation. A performant, lightweight set of optimization algorithms designed for distributed cloud computing are needed that could be used in conjunction with highly performant lightweight TML and DNN-based algorithms described in STADE. Recent advances in Federated Learning have shown promising results in this area [180].

Research in unsupervised representational learning has intensified but supervised, and reinforcement learning remains the focus of industrial research. While neural machine translation has many similarities to the spatiotemporal stream processing domain, such as sequence-to-sequence (seq2seq) modeling, language-translation is fundamentally a supervised learning problem with short sequences. Streaming spatiotemporal anomaly detection is a non-supervised problem, often with long-term temporal and distributed spatial relationships that need to be addressed by novel non-supervised algorithms and architectures such as STADE.

REFERENCES

[1] C. Aggarwal, Outlier Analysis. Second Edition., New York City: Springer, 2017.

[2] A. Adewumi and A. Akinyelu, "A survey of machine-learning and nature-inspired based credit card fraud detection techniques," *International Journal of System Assurance Engineering and Management,* vol. 8, no. 2, pp. 937-953, 2017.

[3] K. Kim, M. Aminanto and H. Tanuwidjaja, Network Intrusion Detection using Deep Learning: A Feature Learning Approach, New York City: Springer, 2018.

[4] D. Weller-Fahy, B. Borghetti and A. Sodemann, "A survey of distance and similarity measures used within network intrusion anomaly detection," *IEEE Communication Surveys & Tutorials,* vol. 17, no. 1, pp. 70-91, 2015.

[5] S. Barbhuiya, Z. Papazachos, P. Kilpatrick and D. Nikolopoulos, "RADS: Real-time anomaly detection system for cloud data centres," *arXiv preprint,* vol. 1811.04481, 2018.

[6] Z. Lu, C. Zhu, X. Liu and X. Sui, "Anomaly detection for virtualized data center via outlier analysis," in *Proceedings of the IEEE 1th International Conference on Networking, Sensing and Control*, Calabria(ITA), 2017.

[7] Wang, Chengwei; Viswanathan, K; Choudur, L; Talwar, V; Satterfield, W; Schwan, K;, "Statistical techniques for online anomaly detection in data centers," Hewlett-Packard, Palo Alto(USA), 2011.

[8] A. Anandakrishnan, S. Kumar, d. Xu and D. Xu, "Anomaly detection in finance: editors' introduction," *Proceedings of Machine Learning Research,* vol. 71, pp. 1-7, 2017.

[9] A. Munawar, P. Vinayavekhin and G. Magistris, "Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, Santa Rosa(USA), 2017.

[10] D. Araya, K. Grolinger, H. Yamany, M. Capretz and G. Bitsuamlak, "Collective contextual anomaly detection framework for smart buildings," in *Proceedings of the International Joint Conference on Neural Networks*, Vancouver(CAN), 2016.

[11] L. Marti, N. Sanchez-Pi, J. Molina and A. Garcia, "Anomaly detection based on sensor data in petroleum industry applications," *Sensors,* vol. 15, no. 2, pp. 2774-2797, 2015.

[12] B. Radford, B. Richardson and S. Davis, "Sequence aggregation rules for anomaly detection in computer network traffic," *arXiv preprint,* vol. 1805.03735, 2018.

[13] S. Varghese and K. Jacob, "Anomaly detection using system call sequence sets," *Journal of Software,* vol. 2, no. 6, pp. 14-21, 2007.

[14] J. Inoue, Y. Yamagata, Y. Chen, C. Poskitt and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *Proceedings of the IEEE International Conference on Data Mining Workshops*, New Orleans(USA), 2017.

[15] C. Leigh, O. Alsibai, R. Hyndman, S. Kandanaarachchi, C. King, J. McGree, C. Neelamraju, J. Strauss, P. Talagala, R. Turner, K. Mengersen and E. Peterson, "A framework for automated anomaly detection in high frequency water-quality data from in situ sensors," *arXiv preprint,* vol. 1810.13076, 2018.

[16] D. Shalyga, P. Filonov and A. Laventyev, "Anomaly detection for water treatment system based on neural network with automatic architecture optimization," *arXiv preprint,* vol. 1807.07282, 2018.

[17] D. Ramotsoela, A. Abu-Hahfouz and G. Hancke, "A survey of anomaly detection in industrial wireless sorsor networks with critical water system infrastructure as a case study," *Sensors,* vol. 18, no. 8, 2018.

[18] L. Vrizlynn and L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proceedings of the Wireless Comminications and Networking Conference*, San Fransciso(USA), 2017.

[19] D. Kwon, H. Kim, J. Kim, S. Suh, I. Kim and K. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing,* pp. 1-13, 2017.

[20] A. Divekar, M. Parekh, V. Salva, R. Mishra and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in *Proceedings of the IEEE 3rd International Conference on Computing, Communication and Security*, Kathmandu(NPL), 2018.

[21] S. Skansi, Introduction to Deep Learning, New York City: Springer, 2018.

[22] K. Cho, B. Merrienboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: encoder-decoder approaches," *arXiv preprint,* vol. 1409.1259, 2014.

[23] M. Yasarm, "Flight anomaly tracking for improved situational awareness: case study of Germanwings flight 9525," in *Proceedings of the Annual Conference on the Prognosis of Health Management Society*, Denver(USA), 2016.

[24] R. Mark, "Five years after MH370 avaiation industry rolling out tech to ensure no plane disappears again," *Forbes Magazine,* 4 March 2019.

[25] A. Jain, B. Verma and J. Rana, "Anomaly intrusion detection techniques: a brief review," *International Journal of Scientific & Engineering Research,* vol. 5, no. 7, 2014.

[26] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection for discrete sequences: a survey," *IEEE Transactions on Knowledge and Data Engineering,* vol. 24, no. 5, pp. 823-839, 2012.

[27] E. Eslami, Y. Choi, Y. Lops and A. Sayeed, "A real-time hourly ozone prediction system using deep convolutional neural networks," *arXiv preprint,* vol. 1901.11079, 2019.

[28] M. Flach, F. Gans, A. Brenning, J. Denzler, M. Reichstein, E. Rodner, S. Bathiany, P. Bodesheim, Y. Guanche, S. Sippel and M. Mahecha, "Multivariate anomaly detection for Earth observations: a comparison of algorithms and feature extraction techniques," *Earth Sytems Dynamics,* vol. 8, pp. 677-696, 2017.

[29] D. Cheng, "The importance of maritime domain awareness for the indo-pacific quad countries," *The Heritage Foundation, No 3392,* 2019.

[30] D. Nguyen, R. Vadaine, G. Hajduch, R. Garello and R. Fablet, "GeotrackNet - A Maritime Anomaly Detector using Probabilistic Neural Network Representation of AIS Tracks and A Contrario Detection," *arXiv preprint,* vol. 1912.00682, 2019.

[31] C. Amariei, P. Diac and E. Onica, "Grand challenge: optimized stage processing for anomaly detection on numerical data streams," in *Proceedings of the 11th ACM International Conference on Distributed and Event-based System.*, Barcelona(ESP), 2017.

[32] A. Adhikari, D. Tax, R. Satta and M. Fath, "Example and feature importance-based explanations for black-box machine learning models," *arXiv preprint,* vol. 1812.09044, 2019.

[33] M. Ribeiro, S. Singh and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACD SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Franscisco(USA), 2016.

[34] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and A. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys,* vol. 51, no. 5, pp. 1-42, 2018.

[35] Y. Bengio, T. Dekeu, N. Rahaman, N. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal and C. Pal, "A Meta-transfer Objective for Learning to Disentangle Causal Mechanisms," *arXiv preprint,* vol. 1901.19012, 2019.

[36] Y. Kou, C. Lu and S. Sirwongwattana, "Survey of fraud detection techniques," in *Proceedings of the 2004 IEEE Intenrational Conference on Networking, Sensing, & Control*, Taipei(TWN), 2004.

[37] D. Li, D. Chen, J. Goh and S. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *arXiv preprint,* vol. 1809.04758, 2018.

[38] D. Li, D. Chen, L. Shi, B. Jin, J. Goh and S. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," *arXiv preprint,* vol. 1901.04997, 2019.

[39] K. Mehrotra, C. Mohan and H. Huang, Anomaly Detection Principles and Alogrithms, New York City: Springer, 2017.

[40] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," in *Proceedings of the 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems*, Singapore(MYS), 2015.

[41] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," PLOS ONE, San Francisco, 2016.

[42] B. Pincombe, "Anomaly detection in time series of graphs using ARMA processes," *ASOR Bulletin,* vol. 24, no. 4, 2005.

[43] E. Burnaev and V. Ishimtev, "Conformalized density and distance-based anomaly detection in timer-series data," *arXiv preprint,* vol. 1608.04585, 2016.

[44] T. Lee, J. Gottschlich, N. Tatbul, E. Metcalf and S. Zdonik, "Greenhouse: A zero-positive machine learning system for time-series anomaly detection," *arXiv preprint,* vol. 1801.03168, 2018.

[45] L. Wei, N. Kumar, V. Lolla, E. Keogh, S. Lonardi and C. Ratanamahatana, "Assumption-free anomaly detection in time series," in *Proceedings of the 17th International Conference on Scientific and Statistical Database Management*, Santa Barbara(USA), 2005.

[46] M. Salechi and L. Rashidi, "A survey on anomaly detection in evolving data," *ACM SIGKDD Explorations Newsletter,* vol. 20, no. 1, pp. 13-23, 2018.

[47] E. Dereszynski and T. Dietterich, "Spatiotemporal models for data anomaly detection in dynamic environmental monitoring campaigns," *ACM Transactions on Sensor Networks,* vol. 8, no. 1, pp. 1-36, 2011.

[48] T. Klerx, M. Anderka, H. Buning and S. Priesterjahn, "Model-based anomaly detection for discrete event systems," in *Proceedings of the IEEE 26th International Conference on tools with Artificial Intelligence*, Limassol(CYP), 2014.

[49] M. Schneider, W. Ertel and F. Ramos, "Expected similarity estimation for large-scale batch and streaming anoamly detection," *Machine Learning,* vol. 105, no. 3, pp. 305-333, 2016.

[50] X. Shi, R. Qiu, X. He, Z. Chu, Z. Ling and H. Yang, "Anomaly detection and location in distribution network: a data-driven approach," *arXiv preprint,* vol. 1801.01669, 2018.

[51] X. Shi, R. Qiu, Z. Ling, F. Yang and X. He, "Spatio-temporal correlation analysis of online monitoring data for anomaly detection in distribution networks," *arXiv preprint,* vol. 1810.08962, 2018.

[52] H. Song, Z. Jiang, A. Men and B. Yang, "A hybrid semi-supervised anomaly detection model for high-dimensional data," *Computational Intelligenece and Neuroscience,* vol. 1, pp. 1-9, 2017.

[53] M. Siddiqui, A. Fern, T. Dietterich and W. Wong, "Sequential feature explanations for anomaly detection," *arXiv preprint,* vol. 1503.00038, 2015.

[54] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, Denver(USA), 1999.

[55] C. Aggarwal, Neural Networks and Deep Learning: A Textbook, New York City: Springer, 2017.

[56] S. Ramaswamy, R. Rastogi and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM Sigmod Record,* vol. 29, no. 2, pp. 427-438, 2000.

[57] M. Breunig, H. Kriegel, R. Ng and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the ACM SIGMOD 2000 International Conference on Management of Data*, Dallas(USA), 2000.

[58] Z. He, X. Xu and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters,* vol. 24, no. 9-10, pp. 1641-1650, 2003.

[59] M. Goldstein and A. Dengel, "Histogram-based outlier score. A fast unsupervised anomaly detection algorithm," in *Poster and Demo Track of the 35th German Conference on Artificial Intelligence*, Saarbrücken(Ger), 2012.

[60] F. Liu, K. Ting and Z. Zhou, "Isolation forest.," in *Proceedings of the International Conference on Data Mining*, Pisa(ITA), 2008.

[61] J. Hardin and D. Rocke, "Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator," *Computational Statistics & Data Analysis,* vol. 44, no. 4, pp. 625-638, 2004.

[62] P. Rousseeuw and K. Van Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics,* vol. 41, no. 3, pp. 212-223, 1999.

[63] M. Amer, M. Goldstein and S. Abdennadher, "Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Dection and Description (ODD'13)*, New York City, 2013.

[64] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks*, Portland(USA), 2003.

[65] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science,* vol. 313, no. 5786, pp. 504-507, 2006.

[66] S. Upadhyaya and K. Singh, "Nearest neighbor-based outlier detection techniques," *International Journal of Computer Trends and Technology,* vol. 3, no. 2, pp. 299-303, 2012.

[67] C. Tsai and C. Lin, "A triangle area based nearest neighbor approach to intrusion detection," *Pattern Recognition,* vol. 43, no. 1, pp. 222-229, 2010.

[68] Y. Zhao, Z. Nasrullah and L. Zheng, "PyOD: A python toolbox for scalable outlier detection," *Jornal of Machine Learning Research,* vol. 20, no. 96, pp. 1-7, 2019.

[69] R. Josefowicz, W. Zaremba and I. Sutskever, "An empirical exploration of recurrent neural network architectures," in *International Conference on Machine Learning*, Paris(Fr), 2015.

[70] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A Survey," *arXiv preprint,* vol. 1901.03407, 2019.

[71] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint,* vol. 1412.6980, no. V9, 2017.

[72] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by backpropaging errors," *Nature,* vol. 323, no. 6088, pp. 533-536, 1986.

[73] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint,* vol. 1609.04747, 2016.

[74] J. Berstra, R. Bardenet, Y. Bengio and B. Kegl, "Algorithms for hyper-parameter optimization," in *Proceedings of the Twenty-Fifth Conference on Neural Information Processing Systems*, Granada(ESP), 2011.

[75] T. Salimans and D. Kingma, "Weight normalization: a simple reparameterization to acclereate training of deep neural networks," in *Proceedings of the Thirtieth Conference on Neural Information Processing Systems*, Barcelona(ESP), 2016.

[76] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *arXiv preprint,* vol. 1502.03167, 2015.

[77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research,* vol. 15, pp. 1929-1958, 2014.

[78] I. Sutskever, J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta(USA), 2013.

[79] I. Goodfellow, "NIPS 2016 Tutorial: Generative adversarial network," *arXiv preprint,* vol. 1701.00160, 2016.

[80] Y. Bengio, A. Courville and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, pp. 1798-1828, 2013.

[81] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge: The MIT Press, 2016.

[82] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," SNU Data Mining Center, Seoul, 2015.

[83] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations*, Banff(CAN), 2014.

[84] B. Barz, E. Rodner, Y. Garcia and J. Denzler, "Detecting regions of maximal divergence for spatio-temporal anoamly detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 41, no. 5, pp. 1088-1101, 2018.

[85] B. Zong, Q. Song, M. Renqiang Min, W. Cheng, C. Lumezanu, D. Cho and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *Proceedings of the Sixth International Conference on Learning Representations*, Vancouver(CAN), 2018.

[86] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Montreal, 2014.

[87] Z. Lipton, J. Berkowitz and C. Elkau, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint,* vol. 1506.00019, 2015.

[88] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions of Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1785, 1997.

[90] S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A field guide to dynamical recurrent neural networks*, IEEE Press, 2001.

[91] R. Pascanau, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the International Conference on Machine Learning*, Atlanta(USA), 2013.

[92] A. Vaswani, N. Shazeer, N. Parmar, J. Uskoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," *arXiv preprint,* vol. 1706.03762, 2017.

[93] K. Cho, B. Merrienboer, D. Bahdanau, H. Schwenk and Y. Benio, "Learning phase representations using RNN encoder-decoder for statistical machine translaiton," in *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*, Doha(QAT), 2014.

[94] F. Chollet, Deep Learning with Python, Shelter Island: Manning Publications Co, 2018.

[95] R. Ariyaluran, F. Habeeb, F. Nasaruddin, A. Gani, A. Targio, E. Ahmed and M. Imran, "Real-time big data processing for anomaly detection: A Survey," *International Journal of Information Management,* vol. 45, pp. 289-307, 2019.

[96] M. Ahmed, A. Mahmood and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications,* vol. 60, pp. 19-31, 2016.

[97] M. Bhuyan, D. Bhattacharyya and J. Kalita, Network traffic anomaly detection and prevention: concepts, techniques, and tools, New York City: Springer, 2017.

[98] Y. Bengio, Learning deep architectures for A.I., Boston: Now Publishers, 2009.

[99] S. Hawkins, H. He, G. Williams and R. Baxter, "Outlier detection using replicator neural networks," in *Proceedings of the Warehousing and Knowledge Discovery 4th International Conference*, Provence(FRA), 2002.

[100] C. Cordero, S. Hauke, M. Muhlhauser and M. Fisher, "Analyzing flow-based anomaly intrusion detection using replicator neural networks," in *Proceedings of the 14th Annual Conference on Privacy, Security and Trust*, Auckland(NZL), 2016.

[101] M. Schreyer, T. Sattarov, D. Borth, A. Dengel and B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoding networks," *arXiv preprint,* vol. 1709.05254, 2017.

[102] J. Chen, S. Sathe, C. Aggarwal and D. Turaga, "Outlier detection with autoencoder ensembles," in *SIAM International Conference on Data Mining*, Houston(USA), 2017.

[103] R. Socher, J. Pennington, E. Huang, A. Ng and C. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Empirical Methods in Natural Language Processing Conference*, Edinburgh(GBR), 2011.

[104] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of the International Conference on Machine Learning Workshop on Unsupervised and Transfer Learning*, Edinburgh(GBR), 2012.

[105] J. Andrews, E. Morton and D. Griffin, "Detecting anomalous data using auto-encoders," *International Journal of Machine Learning and Computing,* vol. 6, no. 1, pp. 21-26, 2016.

[106] R. Aygun and A. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proceedings of the IEEE 4th International Conference on Cyber Security and Cloud Computing*, New York City(USA), 2017.

[107] Y. Ikeda, K. Ishibashi, Y. Nakano, K. Watanabe and R. Kawahara, "Anomaly detection and interpretation using multimodal autoencoder and sparse optimization," *arXiv preprint,* vol. 1812.07136, 2018.

[108] C. Zhou and R. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Mining*, Halifax, Nova Scotia(CAN), 2017.

[109] Y. Chong and Y. Tay, "Abnormal event detection in videos using spatioemporal autoencoder," in *Proceedings of the International Symposium on Neural Networks*, Sapporo(JPN), 2017.

[110] M. Roy, S. Bose, B. Kar, P. Gopalakrishnan and A. Basu, "A stacked autoencoder neural network based automated feature extraction method for anomaly detection in on-line conditioning monitoring," *arXiv preprint,* vol. 1810.08609, 2018.

[111] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, Gold Coast(AUS), 2014.

[112] M. Yousefi-Azar, V. Varadharajan, L. Hamey and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proceedings of the International Joint Conference on Neural Networks*, Anchorage(USA), 2017.

[113] P. Vincent, H. Larochelle, Y. Bengio and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki(FIN), 2008.

[114] G. Dorffnet, "Neural networks for time series processing," *Neural Network World,* vol. 6, pp. 447-468, 1996.

[115] R. Kozma, M. Kitamura, M. Sakuma and Y. Yokoyama, "Anomaly detection by neural network models and statistical time series analysis," in *Proceedings of the 1994 IEEE International Conference on Neural Networks*, Orlando(USA), 1994.

[116] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen and N. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *arXiv preprint,* vol. 1811.08055, 2018.

[117] Z. Rong, D. Shandong, N. Xin and X. Shiguang, "Feedforwared neural network for time series anomaly detection," *arXiv preprint,* vol. 1812.08389, 2018.

[118] R. Chalapathy, A. Menon and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint,* vol. 1802.06360, 2018.

[119] N. Laptev, J. Yosinski, L. Erran and S. Smyl, "Time-series extreme event forecasting with neural networks at Uber," in *Proceedings of the International Conference on Machine Learning Time Series Workshop*, Sydney(AUS), 2017.

[120] Y. Zhang, W. Chen, C. Yeo, C. Lau and B. Lee, "Detecting rumors on online social networks using multi-layer autoencoder," in *Proceedings of the Technology and Engineering Management Conference*, Santa Clara(USA), 2017.

[121] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint,* vol. 1606.05908, 2016.

[122] M. Soelch, J. Bayer, M. Ludersdorfer and P. van der Smagt, "Variational inference for on-line anomaly detection in high dimensional time series.," *arXiv preprint,* vol. 1602.07109, 2016.

[123] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Lui, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," *arXiv preprint,* vol. 1802.03903, 2018.

[124] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Conditional variational autoencoder for prediction and feature reovery applied to intrusion detection in IoT," *Sensors,* vol. 17, no. 9, pp. 1-17, 2017.

[125] Y. Lu and P. Xu, "Anomaly detection for skin disease images using variational autoencoder," *arXiv preprint,* vol. 1807.01349, 2018.

[126] D. Zimmerer, S. Kohl, J. Petersen, F. Isensee and K. Maier-Hein, "Context-encoding variational autoencoder for unsupervised anomaly detection," *arXiv preprint,* vol. 1812.05941, 2018.

[127] D. Kim, H. Yang, M. Chung and S. Cho, "Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things," *arXiv preprint,* vol. 1712.06343, 2017.

[128] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano and L. Benini, "Anomaly detection using autoencoders in high performance computing systems," *arXiv preprint,* vol. 1811.05269, 2018.

[129] J. Walker, C. Doersh, A. Gupta and M. Hebert, "An uncertain future: forecasting from static images using variational autoencoders," in *Proceedings of the European Conference on Computer Vision*, Amsterdam(NLD), 2016.

[130] T. Luo and S. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in WSN in IoT," in *Proceedigns of the IEEE International Conference on Communications*, Kansas City(USA), 2018.

[131] D. Oh and I. Yun, "Residual error-based anomaly detection using auto-encoder in SMD machine sound," *Sensors,* vol. 18, no. 5, pp. 1-14, 2018.

[132] R. Chalapathy, E. Toth and S. Chawla, "Group anomaly detection using deep generative models," *arXiv preprint,* vol. 1804.04876, 2018.

[133] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji and P. Li, "Multidimensional time series anomaly detection: A GRU-based gaussian mixture variational autoencoder approach," in *Proceedings of the 10th Asian Conference on Machine Learning*, Beijing(CHN), 2018.

[134] I. Haloui, J. Gupta and V. Feuillard, "Anomaly detection with Wasserstein GAN," *arXiv preprint,* vol. 1812.02463, 2018.

[135] N. Buitrago, L. Tonnaer, V. Menkovski and D. Mavroeidis, "Anomaly detection for imbalanced datasets with deep generative models," *arXiv preprint,* vol. 1811.00986, 2018.

[136] S. Ger and D. Klabjan, "Autoencoders and generative adversarial networks for anomaly detection for sequences," *arXiv preprint,* vol. 1901.02514, 2019.

[137] M. Kimura and T. Yanagihara, "Anomaly detection using GANs for visual inspection in noisy training data," *arXiv preprint,* vol. 1807.01136, 2018.

[138] S. Lim, Y. Loo, N. Tran, N. Cheung, G. Roig and Y. Elovici, "DOPING: generative data augmentation for unsupervised anomaly detection with GAN," *arXiv preprint,* vol. 1808.07632, 2018.

[139] J. Lima, D. Macedo and C. Zanchettin, "Heartbeat anomaly detection using adversarial oversampling," *arXiv preprint,* vol. 1901.09972, 2019.

[140] T. Matsubara, K. Hama, R. Tachibana and K. Uehara, "Deep generative model using unregularized score for anomaly detection with heterogeneous complexity," *arXiv preprint,* vol. 1807.05800, 2018.

[141] M. Salem, S. Taheri and J. Yuan, "Anomaly generation using generative adversarial networks in host-based intrusion detection," *arXiv preprint,* p. 1812.04697, 2018.

[142] Y. Intrator, G. Katz and A. Shabtai, "MDGAN: Boosting anomaly detection using multi-discriminator generative adversarial networks," *arXiv preprint,* vol. 1810.05221, 2018.

[143] H. Zenati, M. Romain, C. Foo, B. Lecouat and V. Chandrasekhar, "Adversarially learned anomaly detection," *arXiv preprint,* vol. 1812.02288, 2018.

[144] H. Zenati, C. Foo, B. Lecouat, G. Manek and V. Chandrasekhar, "Efficient GAN-based anomaly detection," *arXiv preprint,* vol. 1802.06222, 2018.

[145] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow and B. Frey, "Adversarial autoencoders," in *Proceedings of the International Conference on Language Representations*, San Juan(PRI), 2016.

[146] C. Wang, Y. Zhang and C. Liu, "Anomaly detection via minimum likelihood generative adversarial networks," *arXiv preprint,* vol. 1808.00200, 2018.

[147] J. Elman, "Finding structure in time," *Cognitive Science,* vol. 14, no. 2, pp. 179-211, 1990.

[148] A. Graves, "Geneerating sequences with recurrent neural networks," *arXiv preprint,* vol. 1308.0850, 2013.

[149] A. Karpathy, J. Johnson and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint,* vol. 1506.02078, 2015.

[150] A. de Souza Costa, "Sequence to sequence model for anomaly detection in financial transactions," in *Proceedings of the 33rd International Conference on Machine Learning*, New York City(USA), 2016.

[151] F. Bianchi, E. Maiorino, M. Kampffmeyer, A. Rizzi and R. Jenssen, Neural networks for short-term load forecasting, New York City: Springer, 2017.

[152] A. Brown, A. Tuor, B. Hutchinson and N. Nichols, "Recurrent neural nework attention mechanisms for interpretable system log anomaly detection," *arXiv preprint,* vol. 1803.04967, 2018.

[153] M. Du, F. Li, G. Zheng and V. Srikumar, "Deeplong: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas(USA), 2017.

[154] D. D'Avino, D. Cozzolino, G. Poggi and L. Verdoliva, "Autoencoder with recurrent neural networks for video forgery detection," *Media Watermarking, Security, and Forensics,* vol. 8, pp. 92-99, 2017.

[155] B. Radford, L. Apolonio, A. Trias and J. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint,* vol. 1803.10769, 2018.

[156] R. Malaiya, D. Kwon, J. Kim, S. Suh, H. Kim and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *Proceedings of the International Conference on Computing, Networking and Communications*, Maui(USA), 2018.

[157] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics*, Paris(FRA), 2015.

[158] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," in *International Conference on Machine Learning Anomaly Detection Workshop*, New York City(USA), 2016.

[159] P. Malhotra, L. Vig, G. Shroff and P. Agarwal, "Long short-term memory networks for anomaly detection in time series," in *Proceedings of the European Symposium on Artificial Neural, Networks Computation Intelligence and Machine Learning*, Bruges(BEL), 2014.

[160] L. Bontemps, V. Cao, J. McDermott and N. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural network," *arXiv preprint,* vol. 1703.09752, 2017.

[161] N. Thi, V. Cao and N. Le-Khac, "One-class collective anomaly detection based on long short-term memory recurrent neural networks," *arXiv preprint,* vol. 1802.00324, 2018.

[162] T. Ergen, A. Mirza and S. Kozat, "Unsupervised and semi-supervised anomaly detection with LSTM neural entworks," *arXiv preprint,* vol. 1710.09207, 2017.

[163] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using recurrent neural networks (RNN)," in *Proceedings of the Integrated Communications Navigation and Surveillance Conference*, Herndon(USA), 2016.

[164] D. Park, Y. Hoshi and C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robotics and Automation Letters,* vol. 3, no. 3, pp. 1544-1551, 2018.

[165] Y. Park and I. Yun, "Comparison of RNN encoder-decoder models for anomaly detection," *arXiv preprint,* vol. 1807.06576, 2018.

[166] S. Saurav, P. Malhotra, T. Vishnu, N. Gugulothu, L. Vig, P. Agarwal and G. Shroff, "Online anomaly detection with concept drift adaptation using recurrent neural networks," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, Goa(IND), 2018.

[167] D. Hendrycks, M. Mazeika and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint,* vol. 1812.04606, 2018.

[168] U. Fiore, F. Palmieri, A. Castiglione and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing,* vol. 122, pp. 13-23, 2013.

[169] S. Zhai, Y. Cheng, W. Lu and Z. Zhang, "Deep structured energy-based models for anomaly detection," in *Proceedings of the International Conference on Machine Learning*, New York City(USA), 2016.

[170] J. Kim, J. Yun and H. Kim, "Anomaly Detection for Industrial Control Systems Using Sequence-to-Sequence Neural Networks," *arXiv preprint,* vol. 1911.04831, 2019.

[171] S. Thompson, P. Fergus, C. Chalmers and D. Reilly, "Detection of Obstructive Sleep Apnoea Using Features Extracted from Segmented Time-Series ECG Sginals Using a One Dimensional Convolutional Neural Network," *arXiv preprint,* vol. 2002.00833, 2020.

[172] P. Fergus, C. Chalmers, C. C. Montanez, D. Reilly, P. Lisboa and B. Pineles, "Modelling Segmented Cardiotocography Time-Series Signals Using One-Dimensional Convolutional Neural Netwroks for the Early Detection of Abnormal Birth Outcomes," *arXiv preprint,* vol. 1908.02338, 2019.

[173] S. Russo, A. Disch, F. Blumensaat and K. Villez, "Anomaly Detection using Deep Autoencoders for in-situ Wastewater Systems Monitoring Data," *arXiv preprint,* vol. 2002.03843, 2020.

[174] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D. Inman, "1D Convolutional Neural Networks and Applications - A Survey," *arXiv preprint,* vol. 1905.03554, 2019.

[175] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *Journal of Machine Learning Research,* vol. 9, pp. 2579-2605, 2008.

[176] A. Kossiakoff, W. Sweet, S. Seymour and S. Biemer, Systems Engineering Principles and Practice, 2nd Edition, Hoboken: John Wiley & Sons, Inc., 2011.

[177] M. Stonebraker, U. Cetintemel and S. Zdonik, "The 8 requirements of real-time stream processing," *SIGMOD Record,* vol. 34, no. 4, pp. 42-47, 2005.

[178] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, 2017.

[179] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology,* vol. 10, no. 2, pp. 1-19, 2019.

[180] P. Kairouz and et. al., "Advances and Open Problems in Federated Learning," *arXiv preprint,* vol. 1912.04977, 2019.

[181] S. Ahmad, A. Lavin, S. Purdy and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing,* vol. 262, pp. 134-147, 2017.

[182] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms - the Numenta anomaly benchmark," in *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications*, Miami(USA), 2015.

[183] S. Ahmad and S. Purdy, "Real-time anomaly detection for streaming analytics," *arXiv preprint,* vol. 1607.02480, 2016.

[184] Y. Cui, S. Ahmad and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *arXiv preprint,* vol. 1512.05463, 2015.

[185] S. Guha, N. Mishra, G. Roy and O. Schrijvers, "Robust random cut forest-based anomaly detection on streams," in *Proceedings of the 33rd International Conference on Machine Learning*, New York City(USA), 2016.

[186] S. Lee and H. Kim, "ADSaS: Comprehensive real-time anomaly detection system," *arXiv preprint,* vol. 1811.12634, 2018.

[187] C. Mayer, R. Mayer and M. Abdo, "Streamlearner: Distributed incremental machine learning on event streams: grand challenge," in *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*, Barcelona(ESP), 2017.

[188] J. Pardo, F. Zamora-Martinez and P. Botella-Rocamora, "Online learning algorithm for time-series forecasting suitable for low cost wireless sensor networks nodes," *Sensors,* vol. 15, no. 4, pp. 9277-9304, 2015.

[189] J. Raiyn and T. Toledo, "Real-time road traffic anomaly detection.," *Journal of Transportation Technologies,* vol. 4, no. 3, pp. 256-266, 2014.

[190] E. Khalastchi, G. Kaminka, M. Kalech and R. Lin, "Online anomaly detection in unmanned vehicles," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, Taipei(TWN), 2011.

[191] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proceedings of the 15th IEEE International Conference in Machine Learning and Applications*, Anaheim(USA), 2016.

[192] D. Choudhary, A. Kejariwal and F. Orsini, "One the runtime-efficacy trade-off of anomaly detection techniques for real-time streaming data," *arXiv preprint,* vol. 1710.04735, 2017.

[193] J. Ball, D. Anderson and C. Chan, "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community," *Journal of Applied Remote Sensing,* vol. 11, no. 4, 2017.

[194] S. Budalakoti, A. Srivastava and M. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *IEEE Transactions on Ssytems, man, and Cybernetics, Part C (Applications and Reviews),* vol. 39, no. 1, pp. 101-113, 2009.

[195] M. Hayes and M. Capretz, "Contextual anomaly detection framework for big sensor data," in *Proceedings of the IEEE International Congress on Big Data*, Anchorage(USA)), 2014.

[196] M. Mohammaddii, A. Al-Fugaha, S. Sorour and M. Guizani, "Deep learning for IoT big data and streaming analytics: A Survey," *arXiv preprint,* vol. 1712.04301, 2017.

[197] A. Muallem, S. Shetty, J. Pan, J. Zhao and B. Biswal, "Hoeffding tree algorithms for anomaly dtection in straming datasets: A Survey," *Jornal of Information Security,* vol. 8, no. 4, pp. 339-361, 2017.

[198] S. Xie and Z. Chen, "Anomaly detection and redundancy elimination of big sensor data in internat of things," *arXiv preprint,* vol. 1703.03225, 2017.

[199] M. Yadav, P. Malhotra, L. Vig, K. Sriram and G. Shroff, "ODE - augmented training improves anomaly detction in sensor data from machines," *arXiv preprint,* vol. 1605.01534, 2016.

[200] T. Zaarour, N. Pavlopoulou, S. Hasan, U. Hassan and E. Curry, "Automatic anomaly detection over sliding windows: grand challenge," in *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, Barcelona(SP), 2017.

[201] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Language Representations*, San Diego(USA), 2015.

[202] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho and Y. Bengio, "Attention-based models for speech recognition," in *Proceedings of the 29th Conference on Neural Information Processing Systems*, Montreal(CAN), 2015.

[203] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint,* vol. 1412.3555, 2014.

[204] M. Luong, H. Pham and C. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint,* vol. 1508.04025, 2015.

[205] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le and A. Ng, "Large scale distributed deep networks," in *Proceedings of Advances in Neural Information Processing Systems 25 Conference*, Lake Tahoe(USA), 2012.

[206] K. Chahal, M. Grover and K. Dey, "A hitchhiker's guide on distributed training of deep neural networks," *arXiv preprint,* vol. 1810.11787, 2018.

[207] R. Williams and D. Zipster, "A learning algorithms for continually ruunning fully recurrent neural networks," *Neural Computing,* vol. 1, no. 2, pp. 270-280, 1989.

[208] M. Kastelic and J. Pers, "Building Visual Anomaly Daaset from Satellite Data using ADS-B," in *Proceedings of the 7th OpenSky Workshop*, Zurich, 2019.

[209] M. Schafer, M. Strohmeier, V. Lenders, I. Martinovic and M. Wilhelm, "Bringing up OpenSky: A large-scale ADS-B sensor network for research," in *Proceedings of the 13th international sumposium on information processing in sensor networks*, Berlin, 2014.

[210] A. Tanner and M. Strohmeier, "Anomalies in the Sky: Experiments with traffic densities and airport runway use," in *Proceedings of the 7th OpenSky Workshop*, Zurich, 2019.

[211] S. Mousavi and G. Beroza, "A Machine-Learning Approach for Earthqauke Magnitude Estimation," *arXiv preprint,* vol. 1911.05975, 2019.

[212] R. Aster, "Earthquake forecast for Puerto Rico: Dozens more large aftershocks are likely," *The Conversation,* no. at: https://theconversation.com/earthquake-forecast-for-puerto-rico-dozens-more-large-aftershocks-are-likely-129874, pp. https://theconversation.com/earthquake-forecast-for-puerto-rico-dozens-more-large-aftershocks-are-likely-129874, 14 January 2020.

[213] E. Pavlidou, M. van der Meijde, H. van der Werff and C. Hecker, "Time Series Analysis of Land Surface Temperatures in 20 Earthquake Cases Worldwide," *Remote Sensing,* vol. 11, no. 61, pp. 1-25, 2018.

[214] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, Valletta(MLT), 2010.

[215] R. Yu, H. Qiu, Z. Wen, C. Lin and Y. Liu, "A survey on social media anomaly detection.," *ACM SIGKDD Explorations Newsletter,* vol. 18, no. 1, pp. 1-14, 2016.

[216] D. Savage, X. Zhang, X. Yu, P. Chou and Q. Wang, "Anomaly detection in online social networks," *Social Networks,* vol. 39, pp. 62-70, 2014.

[217] S. Tasoulis, A. Vrahatis, S. Georgakopoulos and V. Plagianakos, "Real time sentiment change detection of twitter data streams," *arXiv preprint,* vol. 1804.00482, 2018.

[218] J. Castellini, V. Poggioni and G. Sorbi, "Fake twitter followers detection by denoising autoencoder," in *Proceedings of the International Conference on Web Intelligence*, Leipzig(DEU), 2017.

[219] P. Werbos, "Backpropagation through time: what is does and how to do it," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1550-1560, 1990.