A Framework for Anomaly Detection under Dynamic and Distributed Scenarios

by

Murugaraj Odiathevar

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Engineering and Computer Science.

Victoria University of Wellington
2021

Abstract

Anomaly Detection is an important aspect of many application domains. It refers to the problem of finding patterns in data that do not conform to expected behaviour. Hence, understanding of expected behaviour well is fundamental to performing effective anomaly detection. However, data profiles constantly evolve in certain domains such as computer networks. In other domains such as traffic monitoring and healthcare, data are distributed and are either too large or there are privacy concerns in transmitting them to a central location. These situations pose a challenge to obtain an accurate understanding of non-anomalous profiles. Changing profiles undermine existing anomaly detection models and make them less effective. Training a robust model with data from multiple sources is also challenging. Moreover, in real world scenarios, it is not apparent how an anomaly detection model can be built to address the problem.

This thesis focuses on the building of a robust anomaly detection system where data profiles evolve and/or are distributed. It proposes a novel Online Offline Framework to separate existing expected behaviour, new possible expected behaviour and anomalies in streaming data. It also addresses the distributed scenario using a theoretically sound fully Bayesian approach. These methods improve performances of anomaly detection systems and work well with biased and uneven data partitions.

The proposed methods are validated using real world data in three different domains. This thesis identifies the implementation difficulties in these domains and produces three novel methodologies to address each of the core anomaly detection problems.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Professor Winston Seah and Professor Marcus Frean for their guidance throughout my PhD journey. I appreciate the valuable insights and suggestions which directed me when I was lost. Without their support, I would not have been able to complete this thesis. Following that, I want to thank Dr Alvin Valera for his feedbacks during the group meetings. I thank the Faculty of Graduate Research and School of Engineering and Computer Science for their support and the doctoral scholarship. I am grateful to Duncan Cameron and Venture Networks and REAANZ for providing data to validate my methods. Consequently, I want to acknowledge Dr Paul Geraghty and Wellington Univentures, and Dr Nick Willis and KiwiNet for the Emerging Innovator award, providing me the opportunity, funding and mentorship to explore real world implementation of the methods developed in my thesis. I would also like to thank and acknowledge two of my colleagues Dr Jakob Pfender and Dr Guiying (Victoria) Huang for proofreading my thesis.

Sincere thanks to my other colleagues: Dr Deepak Singh, João Costa, Kosisochukwu Madukwe, Mazhar Ansari Ardeh, Baligh Al-Helali, Hiroshika Hinduramage, Harisu Abdullahi, Mahdi Abdollahi for the endless conversations which brightened my days and nights at the office.

Last but not least, I would like to thank my good friends, Usha Nadarajan, Shamini Sukumaran, Naira de Garcia and Tamina Beveridge for their support outside of work.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

RE Reconstruction Error.

k**-NN** k-Nearest Neighbours.

Acc Accuracy.

AE AutoEncoder.

AI Artificial Intelligence.

AP Average Precision.

API Application Programming Interface.

AS Autonomous System.

AUC Area Under ROC Curve.

BAE Bayesian AutoEncoder.

BGP Border Gateway Protocol.

BNN Bayesian Neural Network.

BRVFL-AE Bayesian Random Vector Functional Link AutoEncoder.

BS Background Subtraction.

ByB Bayes-by-Backprop.

CAE Convolutional AutoEncoder.

CC Closeness Centrality.

DAE Denoising AutoEncoder.

DC Degree Centrality.

DoS Denial-of-Service.

EP Expectation Propagtion.

FL Federated learning.

FPR False Positive Rate.

GMM Gaussian Mixture Model.

GRT Global Routing Table.

IID Independent and Identically Distributed.

IoT Internet of Things.

IP Internet Protocol.

ISP Internet Service Provider.

KDE Kernel Density Estimation.

LOF Local Outlier Factor.

MAP Maximum-A-Posteriori.

MCMC Markov Chain Monte Carlo.

ML machine learning.

MW-test Mann-Whitney U test.

NN Neural Network.

OCSVM One Class Support Vector Machine.

OD Object Detection.

PCA Principal Component Analysis.

PR Precision-Recall.

Rad-NN Radius Nearest Neighbours.

ROC Receiver Operating Characteristic.

RTT Round-Trip-Time.

RVFL Random Vector Functional Link.

SAE Sparse AutoEncoder.

SDN Software Defined Networking.

SGHMC Stochastic Gradient Hamiltonian Monte-Carlo.

SLFN Single Layer Feed-forward Neural Network.

SNR Signal-to-Noise ratio.

SORT Simple Online and Realtime Tracking.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TPR True Positive Rate.

VAE Variational AutoEncoder.

VI Variational Inference.

WSN Wireless Sensor Network.

Chapter 1

Introduction

Anomaly Detection is an important aspect of many application domains such as network monitoring [17], traffic surveillance and monitoring [181], fraud detection [30], climate monitoring [30], autonomous vehicles [92] and industrial Internet of Things (IoT) [111, 112]. Anomaly detection is about finding patterns in data that do not conform to the expected behaviours [30]. The challenge is to obtain an accurate understanding of expected behaviours. The requirements and scenarios for anomaly detection have changed considerably because of the huge increase in data sources and data volumes [50] brought about by recent trends in technology which include 5G [131], IoT, and Edge Artificial Intelligence (AI). 5G is the next generation cellular network to support demanding services such as mobile broadband, reliable and low latency communications, machine to machine communications, etc [131]. IoT represents the proliferation of devices being able to sense, collect and transmit data over the Internet [70]. Edge AI is the process of pushing AI capabilities of training and inference towards the edge of the network closer to the application or the user [170].

This thesis focuses on two main theoretical scenarios, which pose a challenge for anomaly detection. They are data streams that evolve over time and distributed data. *Concept-drift* is the phenomenon where the values and trends in the data change over time, i.e. when the underlying dis-

tribution of the data changes. The way that data evolve might be systematic or random but this is not an anomaly. However, it affects performance of anomaly detection models. Evolving data is a real-life phenomenon which is seen in many situations and applications [71]. In computer networks for example, Software Defined Networking (SDN) allows one to control the network centrally using software applications which gives rise to concept-drift [212]. The deployment of new types of network applications, application layer protocols or services result in new network session behaviours that can be distinctive, and even normal data displays variations [89, 162, 167]. New legitimate traffic in network data can be new services, new protocols or new profiles of data such as flash crowd data. For example, a host-server sending only images now allows for sending of videos under the same application. These new data are different from the data used to train a model and they are usually and should be detected as anomalous. However, when this happens, legitimate anomalies are missed because the incoming data are considered as false positives.

The second theoretical scenario is distributed data. Traditionally, after data are aggregated from numerous sites and sensors, they are sent to the cloud and are used for analysis or training a model. In this Big Data era, sending data over the network is expensive, especially in wireless networks. They can use most of the bandwidth, hindering other applications or processes. Moreover, a model needs to be trained using the data and deployed quickly back at the edge for certain real-time applications such as in autonomous vehicles, surveillance and predictive maintenance in industrial sites [196]. Furthermore, in certain sectors such as healthcare, transmitting data also raises privacy concerns. Fortunately, with the advent of Edge AI and the advance in technology, computational capabilities are being pushed to the edge of the network [50, 170]. This shift in paradigm can redefine anomaly detection systems to work over networks without transmitting data but rather only certain parameters, thus lowering communication costs and protecting data privacy. Mov-

ing away from having a central server also avoids having a single point of failure. For example, crash detection in traffic surveillance [148], invalid router updates in network connections [103], pedestrian detection in self-driving cars [215], anomalous condition detection in medical electrocardiograms [30], etc. are all mission critical scenarios which require quick decision making and tractability. Models can be now trained closer to the site of the application and this allows for quick deployment and decision making. However, these benefits also come with new constraints. Learning locally at each site on local data is not sufficient to build a robust model especially when the data may vary. The challenge now is to build a global model, similar to the one that would have been built at the cloud with all of the data. The constraint is to avoid sending raw data. Though, the computational capabilities at the edge have improved over the years, it cannot match up to the one at the cloud. This thesis explores this challenge and utilises a Bayesian approach to build a global model for anomaly detection by transmitting only the parameters of the model. Bayesian methods and their rationale are explained further in Sections 2.2 and 4.1.

The definition of "anomaly" depends on the application. For example, in a Wireless Sensor Network (WSN), a malfunctioning sensor can be the anomaly to be detected. In Internet Protocol (IP) traffic, malwares are anomalies to be detected. In video streams of road traffic, crash detection is equivalent to anomaly detection. However, it is not feasible to anticipate all types of anomalies. Malwares may not be the only type of anomalies to detect or crashes may not be the only anomalies of interest. So the space of anomalies is usually too large to be defined. Fortunately, the space defined by *non*-anomalous data is much more manageable. Hence, understanding normal behaviour is vital to detecting other anomalous behaviours both existing known and unknown new ones. In this thesis, normal data mean non-anomalous data. This thesis explores many existing machine learning models and gives only an overview of each unless it is specifically used for achieving the objectives. Details of each method can be found in the wider

Internet and in the literature [24].

In practice, successful anomaly detection requires a robust understanding of the application domain, which brings its own challenges. Typically, there are many steps involved to capture the right information before anomaly detection can even take place. To address the whole chain from a domain-specific issue to a robust solution, this thesis looks at three generic real world domains: systems monitoring, network graphs monitoring and surveillance monitoring. In monitoring systems such as industrial equipment and machines, operations in data centers, solar charges at power stations, etc., the challenge lies in identifying the right features to build a useful anomaly detection model. Capturing network connection information is vital for detecting anomalous in networked scenarios such as Internet traffic routes, connections between sensors in WSNs and power grids. Extracting useful information quickly is essential for training an anomaly detection model in surveillance monitoring. In doing so, this thesis introduces three novel methods to build an anomaly detection system for each of the mentioned domains.

1.1 Motivation

Dynamic scenarios occur in every application of anomaly detection as things are always changing. With the proliferation of data with 5G and IoT, the need to address issues that occur in a dynamic scenario intensifies. However, the study of dynamic scenarios have not been thoroughly addressed in anomaly detection literature. Methods have been focused on building anomaly detection models which can generalise to new data instead of viewing dynamic scenarios uniquely. This thesis is the first to move the study of anomaly detection in this direction.

Distributed scenarios are becoming more common in recent years and will be expected to become more necessary in the future. This can be seen by the increase in the volume of research focused on Edge AI [150, 209].

Thus, being able to detect anomalous data at the edge quickly with a relatively simple model is essential because computational and memory capabilities in edge devices are generally low due to hardware constraints. Hence, this is a challenging problem which this thesis aims to tackle.

1.1.1 Problems with current methods

There is a vast literature as will be shown in Chapter 2 on anomaly detection methods to improve detection rates and to reduce false positive rates [24]. Detection refers to correctly identifying an anomaly. False positives refer to legitimate data incorrectly identified as anomalous. In this context, methods fall into the categories known as supervised, unsupervised or semi-supervised.

Supervised methods contain examples of both anomalous and non-anomalous classes during training. These are also called *Signature-based* methods [24]. Although supervised methods perform well in detecting known anomalies [24], they typically fail at detecting new anomalies. Furthermore, obtaining and labelling samples of anomalous data is very time consuming [30].

Unsupervised methods assume that normal data points are majority and perform outlier detection [67]. This assumption is not valid in a number of scenarios. For example, Denial-of-Service (DoS) attacks are anomalies in the context of network security and such data points occur together with similar profile. Furthermore, with high dimensional data, outlier detection rarely performs well because of the curse of dimensionality [3].

Semi-supervised methods which only use normal class during training are also known as *Behaviour-based* methods. Behaviour-based methods lose their validity when encountering new variants of normal data. For example in network monitoring, the provision of a new service, a change in protocols, or addition of new sensors all lead to a new "normal". A successful Behaviour-based model identifies existing normal data and gen-

eralises to new normal data well. Some methods in machine learning literature to achieve generalisation are the Denoising AutoEncoder (AE) (DAE) [19], Dropout [178] and Early Stopping [152]. However, verifying whether they have generalised to unseen or evolved data is a nearly impossible task. In the literature, generalisation ability is evaluated on a hitherto unseen test set. However adversarial examples may be found [28,82]. One cannot conclusively state how the machine learning model generalises to new data. As such, one cannot be confident that the model will distinguish all normal data from anomalies, both current and future. In critical systems, this is not a risk worth taking, and suppresses the adoption of machine learning based methods in the real world, where a high cost is paid for errors [176].

1.1.2 Evolving data

One way to address evolving data is using online methods which train and perform on live streaming data [24]. Although online methods capture concept-drift more effectively [58], they are not effective when anomalies happen in clusters as they are mainly outlier detection methods. Existing anomaly detection methods can be categorised as online or offline. Offline models are traditional models trained with data offline and deployed in the system for anomaly detection. Many existing systems, such as firewalls, which can be either Signature-based or Behaviour-based methods, fall under this category. They can be optimised to pick out patterns in high dimensional data that are not easily recognisable by humans but they suffer from lengthy training times and require updates with new data profiles. Online methods on the other hand suffer from lack of labelled training data. In this train of thought, this thesis proposes a novel Hybrid Online Offline Framework to obtain the benefits of both online and offline models while addressing each of its limitations.

1.1.3 Heterogeneously Distributed Data

Current methods are also not suited for distributed training at the edge where the data are distributed in many locations. Distributed training methods for neural networks such as synchronous or asynchronous stochastic gradient descents [29] have high computational complexities and long training times which may not be feasible in edge networks. Some studies [186] also explore ensemble methods where each site trains a model using its own data and transmits the parameters to other sites. The models are deployed as an ensemble during deployment. Other studies [112] perform aggregation of the parameters or combine clusters to build a global model. However, such ad-hoc averaging methods are not suitable when the data are heterogeneously distributed over many sites. Here heterogeneously distributed data refers to data which are from different distributions and partitioned in an uneven and biased manner. Existing studies do not account for such a scenario in their evaluations in comparison to this thesis. More details are provided in Chapter 4.

1.2 Research Question

Designing an anomaly detection system under evolving and distributed data remains as an open research problem as prior studies on anomaly detection have not considered them. Thus, this thesis attempts to address the following research question.

How can anomaly detection take into consideration evolving and heterogeneously distributed data?

Moreover, there are many steps involved to set-up real world data for effective anomaly detection. Whilst attempting to answer the above research question, this thesis also considers the difficulty of handling real world domains. A supplementary question is as follows.

How to represent data in different domains for anomaly detection?

This is another aspect which is not addressed in the existing literature. Useful information is unique to the problem and so there is no one right answer. This thesis attempts to address the following question in three generic domains of systems monitoring [140], network graphs monitoring and surveillance monitoring in video streams. This thesis also validates the novel methods developed in these scenarios.

1.3 Research Objectives

The main research question can be answered in parts. The research objectives of this thesis are as follows.

(A) **Design an anomaly detection system that is robust under evolving streaming data.**
Online methods from the literature already capture concept-drift. However, to train them to perform well, they require dimensionality reduction and a good reference of normal data which offline methods can achieve. This thesis attempts to design a framework combining the strengths of an online and an offline model to achieve the objective.

(B) **Develop an anomaly detection model that can be trained on heterogeneously distributed data.**
This objective naturally comes with the constraint of not allowing the data to be aggregated in one location or site. Sufficient statistics or parameter values can be sent to each site but not the entire dataset. This thesis approaches this question by building an anomaly detection model using Bayesian inference which is more robust compared to other methods.

(C) **Explore methods to prepare data for anomaly detection in real world domains while validating the methods from objectives (A) and (B).**

The literature for anomaly detection is focused on methods while ignoring this important aspect. The challenge is to obtain useful information in the context to build-up an anomaly detection system. Hence, many of the methods are not deployed in the real world. As part of the validation, this thesis shows and explains the many steps involved in the collection and preparation of real world data to perform effective anomaly detection in each of the domains.

1.4 Research Contributions

This thesis makes the following major contributions:

1. *A novel Online Offline framework to build a anomaly detection system that is robust under evolving data*
   This framework can be utilised for both Signature-based and Behaviour-Based methods. Under evolving data, an independent online and offline model can be improved using this framework to achieve better detection rate and lower false positive rate. This framework can also be used to mitigate inaccurate generalisation and allows detection of new normal data. Thus, it addresses objectives (A). This contribution has been published in [141,142]. This contribution has also been patented [143].

2. *A novel approach using Bayesian methods to perform distributed training for anomaly detection under heterogeneously distributed data.*
   This approach requires only the parameters of the model to be sent over the communication network. The fully Bayesian approach builds a Bayesian Neural Network (BNN) and its parameters are aggregated using Expectation Propagtion (EP). The use of EP also requires the use of Variational Inference (VI) to train the BNN. In a Single Layer Feed-forward Neural Network (SLFN), a closed form solution is possible. The training and aggregation is quick and the method is

robust under heterogeneously distributed data as compared to other methods in the literature.

3. *Methods to obtain useful information in real world domains for anomaly detection.*
   This thesis addresses the challenges of three real world domains and presents three sub-contributions.

   (a) The first domain is on systems monitoring and an example scenario of an antenna site at an Internet Service Provider (ISP) is considered. Venture Networks, a rural ISP located in the Horowhenua District of New Zealand provides Internet connectivity to local farms, and has several sites across the region. The problem particular to this domain is in identifying the right features to be able to build a useful anomaly detection model. This thesis designs a novel Bottom-up approach and an iterative procedure to address the problem of feature identification for systems monitoring. This contribution is published in [140].

   (b) The second domain is on monitoring network graphs and the considered scenario is connections between core routers defined by the Border Gateway Protocol (BGP) protocol. REANNZ [157] monitors the connections between research institutions within and in and out of New Zealand. They are also aware of how the core routers of the world are connected. Sometimes there have been misconfigurations or malicious attempts at breaking these connections which lead to Internet downtime through the BGP protocol [5]. This thesis implements a novel method by building a network graph and using its centrality values as input to train an anomaly detection model to address BGP anomaly detection.

   (c) The third domain is on surveillance monitoring. For this domain, the scenario of detecting anomalies such as accidents, stalled vehicles, etc. based on video feeds from cameras at high-

ways is considered. To detect an accident on the road, this thesis uses different methods to extract object and motion information and builds a novel pipeline which leads to training a relatively simple and deployable anomaly detection model.

1.5 Awards, Patents and Publications

- Murugaraj Odiathevar, "KiwiNet Emerging Innovator Award 2020", *https://kiwinet.org.nz/EmergingInnovatorProgramme*, 2020 [139]
- Murugaraj Odiathevar, Winston KG Seah, Marcus Frean, "Patent on Hybrid Machine Learning System", *https://www.ipaustralia.gov.au/patents*, Australian IP application number: 2019900788, 2019 [143]
- Murugaraj Odiathevar, Winston KG Seah, Marcus Frean, and Alvin Valera, "Patent on Hybrid Machine Learning System and Method", *https://patentscope.wipo.int/search/en/search.jsf*, World Intellectual Property Organization (WIPO) - application number: PCT/NZ2020/050025, 2020 [144]
- Odiathevar, Murugaraj, Winston K. G. Seah, and Marcus Frean. "A hybrid online offline system for network anomaly detection." *28th International Conference on Computer Communication and Networks (ICCCN)*, 2019. [141]
- Odiathevar, Murugaraj, et al. "An Online Offline Framework for Anomaly Scoring and Detecting New Traffic in Network Streams." *IEEE Transactions on Knowledge and Data Engineering (2021)*. [142]
- Odiathevar, Murugaraj, et al. "Humans Learning from Machines: Data Science Meets Network Management." Invited Paper, *International Conference on COMmunication Systems and NETworkS (COMSNETS)*, 2021. [140]

1.6 Thesis Structure

This thesis is structured as follows. Chapter 2 reviews the literature and is broken into Offline models, Online models, Bayesian methods and distributed model training and existing distributed anomaly detection methods. Chapter 3 introduces the Hybrid Online Offline Framework, evaluates it on public datasets and shows an example for binary classification. Chapter 4 shows how to use Expectation Propagation (EP) for distributed training on a Single Layer Feed-Forward Neural Network (SLFN) and effectiveness on a deep neural network as well. This is evaluated on various public datasets. Chapter 5 studies the three real world domains for anomaly detection and shows the use of the approaches from the earlier chapters after obtaining useful information in each context. Chapter 6 summarises and talks about future directions to extend this thesis.

Chapter 2

Background and Literature Review

This chapter reviews related work that form the background and motivation for the rest of this thesis. This chapter begins with the different types of anomaly detection methods and highlights their strengths and weaknesses. Subsequently, distributed methods for training a model are reviewed. Finally, the public datasets and evaluation measures used in this thesis are introduced.

2.1 Anomaly Detection Models

There are many papers on anomaly detection in the literature and they are categorized here as either online, offline or a combination. A comprehensive review on the many techniques used can also be found in [24].

2.1.1 Introduction

This subsection briefly introduces some well known methods in the literature which are also explored in this thesis. The first method is the Principal Component Analysis (PCA). Given dataset of d-dimensional vectors,

PCA finds a reduced dimensional hyperplane that minimises the square of the projection error over the remaining dimensions. In other words, the subspace defined by a linear combination of the principal components provides a coordinate system to represent the data with little loss [3]. Data points with large projection distances to this subspace are viewed as outliers. This approach assumes that the relationship between observed variables is linear, which is usually too simplistic.

The Support Vector Machine (SVM) [40] on the other hand, maps the data to a higher dimensional space using the *kernel trick* to draw a hyperplane to separate two classes in the data. The one-class variant known as the One Class Support Vector Machine (OCSVM) [165] is trained only on normal data and aims to separate the data from the origin in the same manner. Data points that fall closer to the origin are then identified as outliers. The decision boundary is defined using a few of the training data points which are called support vectors.

Another common approach is using the Gaussian Mixture Model (GMM). This probabilistic method is relatively simple yet effective. It assumes that the data is the output of a generative process in which each point belongs to one of a few Gaussian clusters whose parameters are learnt from data [3]. Data points that have a low probability of being generated by any Gaussian cluster can be deemed as outliers. A GMM with enough mixtures can model any distribution [68]. The Kernel Density Estimation (KDE) method is a non-parametric variant of the GMM to estimate the probability density function of the data. A Gaussian or other kernel density is placed on each data point and one obtains a smooth density for the data.

Some other common density based estimators are k-Nearest Neighbours (k-NN) [9], Radius Nearest Neighbours (Rad-NN) and Local Outlier Factor (LOF) [26]. k-NN involves holding the training dataset in memory and calculating the distance of the nearest k data points to a new vector of observations. Larger distances denote outliers. Similarly, the Rad-NN

involves counting the number of nearest data points within a specified radius, and low numbers indicate outliers. The LOF is able to adjust for variations in different local densities. This attribute allows outlier detection of data which have small distances to a highly dense neighbourhood, compared to normal data which may have larger distances but in a sparse neighbourhood.

Most of the above are susceptible to the curse of dimensionality which states that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables [14]. Thus, for statistically significant results, exponentially more data is required for training a model. This becomes a problem in online models when data is available in small batches yet many features need to be captured. In view of this, deep learning methods have become more favourable [25] and in particular deep AutoEncoders (AEs) for anomaly detection [11, 27, 130, 136, 207, 207]. An AE consists of an encoder network, a latent layer and a decoder network. The encoder maps the input data into the latent layer of lower dimension, and the decoder reconstructs them again. This is also called an undercomplete AE, and is shown in Figure 2.1. The weights of both maps are determined using the data during training phase. Training an undercomplete AE forces the AE to retain important variations that are required to reconstruct the input. As a result, the latent layer provides a succinct representation of high dimensional data [130]. Where the AE is trained with only normal data, the associated Reconstruction Error (RE) can be used as an anomaly score [51, 68]. The underlying assumption is that normal data can be reconstructed (low RE) while anomalous data will have higher RE.

The AE learns the distribution of the normal data. The RE measure can then be viewed as a vertical distance from the center/peak of the distribution. Points with a larger RE tend to mean that the data point is far from the centre. A 2 dimensional density plot is depicted in Figure 2.2. Points in each contour line have the same RE. Points with smaller RE are closer

to the center of distribution of the training data and likewise, a larger RE denotes that the data point is further from the centre.

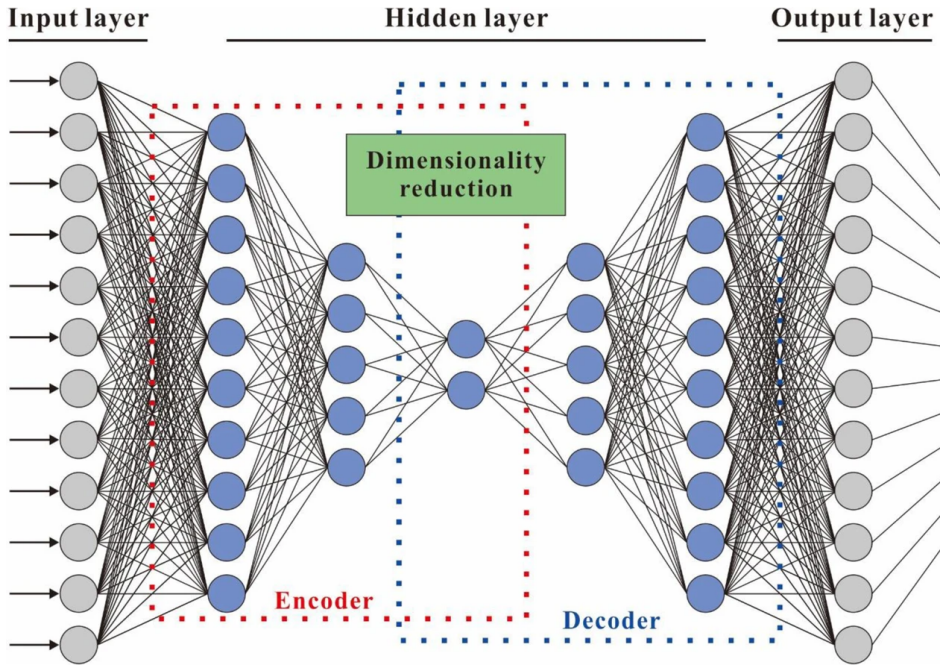


Figure 2.1: A Neural Network AutoEncoder (AE) Model. Lines are weights and each node (circle) is a non-linearity applied to a weighted sum. [128]

Let X denote the normal data and $\mathbf{w}$ denote the parameters of the AE. In essence, an AE is trained to determine $\mathbf{w}$ of a function f such that $f_{\mathbf{w}}(X) = X$. Training an undercomplete AE prevents it from learning the identity function. For a test data point $\hat{\mathbf{x}}$, the RE or the anomaly score is the $L2$ norm.

$$RE = \|f_{\mathbf{w}}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}\|_2 \tag{2.1}$$

There are many variants of the AE such as the Denoising AutoEncoder (DAE) and Variational AutoEncoder (VAE). To extract important features, the DAE corrupts the input data by adding noise and trains by minimising RE with respect to the original inputs [191]. The VAE is a generative

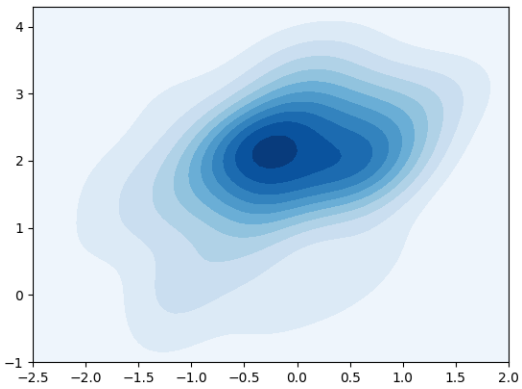


Figure 2.2: Kernel density plot of 50 random points drawn from a 2-dimensional Gaussian distribution with mean (0,2) and covariance [(1,0.5),(0.5,1)]

model trained by minimising RE and the Kullback-Liebler(KL) divergence between the encoder's distribution and the multivariate Gaussian distribution which defines a prior on the latent space [95]. More insights into these methods are provided in Chapter 3.

In the literature, one can find other types of neural networks as well. The Random Vector Functional Link (RVFL) is an example of a Single Layer Feed-forward Neural Network (SLFN) [81]. The weights between the input and the hidden layer are randomly set and remain fixed. Only the weights between the hidden layer and the output layer are trained. The number of nodes in the hidden layer is many times more than the number of input nodes. Like other neural nets, RVFLs can also be formed into AE. In some studies, this is also a form of an Extreme Learning Machine (ELM) as shown in Figure 2.3 [184]. This thesis will make use of such a RVFL-AE, and its Bayesian variant. The rationale is further explained in Chapter 4.

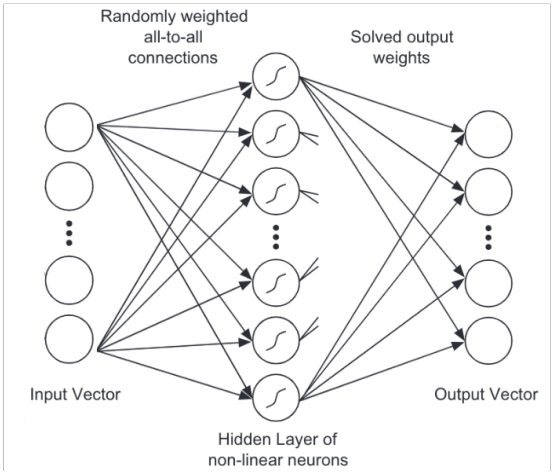


Figure 2.3: A Random Vector Functional Link AutoEncoder (RVFL-AE) is a form of an Extreme Learning Machine when the output and input vectors are the same. [184]

2.1.2 Offline Models

This thesis defines offline models as those trained offline with static data. Because, these usually have long training times and require a large amount of data before they can perform well. After training, the model is then deployed in a system to detect anomalies as and when they occur. Most offline models are supervised or semi-supervised (one-class), meaning some labelled data of one or both classes are available. In most cases, this means all the training data is normal.

Some recent offline methods are OCSVM and Kernel PCA [193], outlier Dirichlet mixture model [121], Affinity Propogation with k-means clustering [195] and LOF and global k-NN [7]. These methods first extract profiles that describe the data, and then use these to determine thresholds or boundaries for normal data. However, in high dimensions, performance drops as a result of the curse of dimensionality, and so finding succinct representations in an unsupervised manner becomes important.

Moustafa *et al.* [121] reduce dimensionality through a simple linear

transformation, namely PCA, but non-linear transformations based on neural nets such as AE have found wide application.

Various Deep Neural Nets (DNN) and their learning algorithms such as Sparse AutoEncoder (SAE)s, DAEs, Contractive AEs, Convolutional AutoEncoder (CAE)s, VAEs, Deep Convolutional Neural Networks (DCNN), Long Short Term Memory (LSTM), Deep Belief Nets (DBN) have been explored in the context of anomaly detection [11, 27, 51, 93, 130, 136, 207, 207]. Erfani *et al.* [51] utilise a linear OCSVM on the latent layer of a Deep Belief Net (DBN) for anomaly detection. Cao *et al.* [27] introduce new regularizers to AE and VAE to force normal data into a tight area centered at the origin in the latent layer of the AE. Aygun *et al.* [11] train an AE and DAE and determine the threshold (for the RE to classify data as anomalous or normal) stochastically. Kim *et al.* [93] merge five Deep Neural Networks (DNN) trained with different sets of features into one DNN. These methods are trained without supervision (ie. targets) to learn salient features of the data [126].

However, these models are unable to incorporate new data in a live stream or dynamic scenario. After training, the weights of these models are fixed. Updating the weights with just the new data would not work because neural networks suffer from a phenomenon called *Catastrophic Forgetting* [96]: the neural network forgets what it learnt of previously. Thus, it needs to be retrained with all of the (old and new) data.

Yin *et al.* [213] propose a deep learning approach using Recurrent Neural Networks (RNN). The authors experimented with different numbers of hidden nodes and varying learning rates. RNNs have more parameters and are able to capture the temporal characteristics of recent data in a stream but like all Neural Network (NN), they still suffer from Catastrophic Forgetting. This method also assumes that the data are temporally related.

Nguyen *et al.* [136] and Xu *et al.* [207] use VAE to detect anomalies. Nguyen *et al.* [136] employ VAE's gradients to cluster the anomalies and

analyse them. The above mentioned methods with the exception of [136, 207] are not evaluated on streaming data. Moreover, these models also need appropriate regularisation in order to be able to score new data correctly but this is problematic as there is no way to ascertain whether the trained model can generalise correctly until after the fact. Even with optimising the model based on a hold-out validation dataset is not sufficient because one cannot be certain that the hold-out dataset captures all possible variations. This becomes apparent when the number of false positives starts to increase in live streams.

The above models are Behaviour-based methods which can detect new anomalous behaviours. Signature-based methods are only able to detect the anomalous profiles used in training. One example of the latter is by Sun *et al.* [182] who firstly detect one category of network traffic using Gradient Boosting Decision Tree. Upon detection, of a specific class of anomaly, the authors use k-NN to classify into subclasses. To detect other types of network traffic, the authors use a stacking ensemble of six classifiers.

2.1.3 Online Models

Online models provide a natural approach to address evolving data. Online models are trained with new data as they arrive, typically in batches. These models are able to capture concept drift [58], by continuously updating the thresholds which distinguish normal from anomalies. Most models work with some form of "knowledge retention" rate, used to balance past versus new information. They are either trained on normal data (semi-supervised or one-class) [135] or are completely unsupervised (outlier detection [67]) [36, 46, 206].

In this space, there are many different methods. Some examples include Local Outlier Factor (LOF) [161], bilateral Principal Component Analysis (PCA) [204], global LOF combined with local subspace outlier detec-

tion in the global space [189] and Kernel Density Estimation (KDE) [72]. Unfortunately, these methods are adversely affected by the curse of dimensionality, because data in high dimensions are sparse and most points end up as "outliers" [4].

Alrawashdeh *et al.* [8] propose a fast activation function to increase convergence speed and accuracy for deep learning in real time. However it is difficult to optimise neural networks with only few iterations. As such, an optimal deep learning model can only be trained offline, which renders it unable to adapt rapidly to new data. Van *et al.* [189] maps high dimensional data to a lower dimension using local subspaces. Huang *et al.* [79] uses Self-Organising Maps (SOM) to reduce dimensions and use a SLFN to learn majority patterns under a dynamically changing environment. A Single Layer Feedforward Neural Network (SLFN) can be trained quickly, which is important in many scenarios; and though it may not learn succinct representations, it does provide a model for anomaly detection. Upon dimensionality reduction, some offline methods such as [121], [195] can also be invoked in an online fashion. Using deep learning to reduce the dimensionality can also improve these methods [51]. This approach has not yet been explored for online models on streaming data. This is one of the aspects of the Online Offline Framework proposed in this thesis.

Another issue with some online models is the difficulty of obtaining ground truth continuously in a live stream with new behaviours. For instance, Lakhina *et al.* [101, 102] perform PCA to reduce dimensions and look for high residuals to detect volume anomalies in origin-destination flows and Nevat *et al.* [135] uses Generalised Likelihood Ratio Test (GLRT) to test the transition matrix of the state-path of TCP traffic flows against that of normal TCP flows. These are semi-supervised models which require normal data for training. The Online Offline Framework addresses this by using the offline model to provide a bias, aiding in the performance of the online model.

Naturally, the lack of labelled data leads to the use of unsupervised methods. One group of unsupervised methods is clustering. Chenaghlau *et al.* [36] perform batch clustering using Density Based Spatial Clusting Application with Noise (DBSCAN). Dromard *et al.* [46] use Density Grid based clustering where they separate the space into grids and bucket data points. The authors use a time sliding window to update the feature space partition. Bigdeli *et al.* [18] experiment with a two-layer cluster based approach by representing each cluster by a Gaussian Mixture Model (GMM). New instances are collected in batches and checked for similarity with existing clusters. Clusters are merged or new ones are generated. Online clustering methods rely on the assumption that normal data clusters, while anomalous data do not. The data points which do not cluster (outliers) are marked as anomalies. However, anomalous data such as DoS (where many data points show similar profile and occur in large quantities) tend to cluster, and hence evade this approach. Thus, it is better to have a model to provide a baseline of what is normal data. In fact, clustering algorithms are better suited at detecting clusters rather than the presence of outliers. This thesis shall use a clustering algorithm to detect new "possibly normal" data instead of outliers.

A number of online models in the literature attempt to retain existing knowledge to some extent, by updating parameters such as cluster means, covariances or thresholds with new data [18, 36, 46, 79, 206]. These are denoted as *incremental learning* in this thesis.

In most data streams and most of the time, it is reasonable to assume that the bulk of the data is normal. However, when anomalies occur in large numbers such as DoS attacks or when there are measurement errors due to malfunctioning devices, having a purely online model fails. Updating thresholds or decision boundaries in such scenarios based on inliers will prove detrimental because it will result in more false negatives in future. On the whole, online models are appropriate for streaming data, so long as the curse of dimensionality is addressed and some guidance in

the form of appropriate inliers or expected normal data is given. Each of the above outlier detection or novelty detection models can be used in the Online Offline framework proposed in this thesis.

Clustering methods will be reserved for detecting new possibly normal data. Detecting new normal data has not be addressed in the literature. These data may be classified as normal or anomalous depending on the model's regularisation. The framework in this thesis shows that these "new normal" should be classified as anomalies because they are different from existing normal data. And using clustering, they can be picked out from the sea of anomalous data points.

2.1.4 Combination methods

There are a few studies in the literature which combine offline and online methods. Existing methods consist of an offline step to select necessary features in a supervised or an unsupervised manner, followed by online learning. Hence, these can easily fall under the banner of online methods.

Su *et al.* [179, 180] use a genetic algorithm to determine the weights for different features (offline aspect). The authors improve k-NN based classifiers using clustering to identify DoS attacks. Recursively, clusters are split based on distances between points to reduce average dispersion. The final resulting centroids are used instead of individual instances for k-NN classification.

Alaei *et al.* [6] preprocess the data to select features for each type of attack in the data set (offline aspect). A Naïve Bayes module incrementally predicts the probability that the new point belongs to one of the attack classes. The authors employ active learning (online aspect) where a random threshold is selected, the labels for instances near this threshold are queried, and the classifiers are updated. The point is classified according to maximum certainty of the classifiers.

An incremental k-NN-SVM method is explored by Xu *et al.* [208]. The

data in batches are clustered using K-Means clustering (online aspect). The cluster centers are classified using k-NN from a database of points (offline aspect). All points in the cluster are then labelled with the label of the cluster center. If the k neighbors do not have the same label, an SVM is trained on the neighbors before classification.

Fischer *et al.* [58] combine an online and offline model for image classification. They use an offline Localised Generalised Matrix Learning Vector Quantization (LGMLVQ) and online incremental classifiers such as incremental online LVQ and incremental SVM with a reliability measure for each classifier and choose the output of the more reliable classifier. This is not used for data streams.

The above are Signature-based methods. Determining the right features using anomalous data falls under Signature-based approach as well as it would determine the important features for those anomalies only.

2.1.5 Summary

Table 2.1: Summary of related work

	Online	**Offline**	**Combination**
Signature -based		[182]	[179,180] [6,208]
Behaviour -based	[8,72,161,189,204] [18,36,46,101,102,135]	[7,121,130,193,195] [11,27,51,93,136,207]	

Table 2.1 gives a summary of the approaches reviewed. It is apparent that an online offline combination method for behaviour based anomaly detection is lacking. The literature also does not distinguish between anomalies and possibly new normal data. This is an important segregation for systems. This thesis is the first to study this.

2.2 Bayesian Approaches and Distributed Implementations

For a model, Bayesian theory paves a way to make inferences about parameters w, given data X. This is commonly known as the posterior distribution, denoted $P(w|X)$. Instead of attributing one value, Bayesian inference allows to attribute a probability distribution to each parameter. For example, the parameters of neural networks are the weights connecting each neuron. If each parameter is independent of one another, they can each be viewed as a one-dimensional Gaussian with a mean and a standard deviation instead of one value. The mode of the posterior also known as the Maximum-A-Posteriori (MAP) estimate provides a one value estimate for the parameters if required. In the Gaussian case, this would be the value of the mean. A brief introduction to the Bayesian approach for ridge regression which is used to build the proposed method in Chapter 4 is given in the appendix C [20, 164].

Bayesian approaches have their benefits as follows [20, 64, 198].

- They allow confidence or uncertainty of the model's output to be computed.
- They provide a natural mechanism to cope with insufficient data.
- They allow prior beliefs to play a role.
- A Bayesian approach to neural networks can potentially avoid some of the pitfalls of stochastic optimization [113].

These come with additional costs:

- High computational cost, especially in models with a large number of parameters.
- There is no clear method to select a prior.

- Longer training times if no closed form solution is available.
- Lack of tractability if no closed form solution is available.

2.2.1 Bayesian Inference Methods

Given the advantages of neural networks in anomaly detection and the benefits of a Bayesian approach, it is natural to consider it in this thesis. The aim is to obtain the posterior distribution of the weights of the neural network. A fully Bayesian treatment of deep neural networks is computationally intractable because of the integration over all weights of the marginal likelihood term and so, a closed form solution is not readily available [22]. For an in-depth formulation, readers are referred to Blei *et al.* [22]. In this subsection, some methods which perform approximate inferences are reviewed.

The first study in training a BNN can be found in Neal [133]. The most well-known method is using the Metropolis Hastings algorithm [37], a form of Markov Chain Monte Carlo (MCMC). The Metropolis Hastings algorithm generates the Markov Chain by accepting and rejecting generated samples and it moves the sample distribution closer to the posterior distribution. MCMC has been shown to work as the number of samples approach infinity. Even when MCMC convergences, it is not easy to determine the number of steps required for convergence [64].

Over the years, this method has been modified and improved to achieve faster convergence. Hamiltonian Monte Carlo uses second order variables and fictitious momentum to drive the sampling closer to the posterior [132]. This method has been improved upon by Chen *et al.* [34] and further optimized using BOHAMIANN by Springenberg *et al.* [177]. Another approach is to use Stochastic Gradient Langevin Dynamics [201].

Apart from MCMC, the posterior can also be approximated by minimising the Kullback-Libler (KL) divergence [100], $\mathrm{KL}[q(w)||P(w|X)]$ between the posterior and an approximate distribution, $q(w)$ such as a Gaus-

sian. This is commonly known as Variational Inference (VI) [20]. Expectation Propagation (EP) instead minimises the reverse KL divergence, KL$[P(w|X)||q(w)]$ [120]. For more details on KL divergence, readers are referred to Bishop [20]. Classical EP was originally designed to incorporate data pointwise into the posterior, although the same idea can be used with data partitioned into subsets [190]. This thesis uses EP to perform distributed training of a model in Chapter 4.

Gal *et al.* [61] and Kingma *et al.* [94] use dropout to perform Bayesian inference. Blundell *et al.* [23] call their method to train a BNN *Bayes by Backprop* (BbB) and Hernández-Lobato *et al.* [75] call theirs probabilistic backpropagation. BbB performs VI via a local reparameterisation trick [94]. Probabilistic backpropagation uses Assumed Density Filtering (ADF) [120] to update the posterior. For more details, readers are referred to the respective studies [23, 75]. There are a number of methods to perform VI but they are not as effective when the data are heterogeneously distributed as will be shown in Section 4.3. EP is a generalised version of ADF (iterative ADF) where it loops over the factors of the posterior multiple times until convergence. EP is described in more detail in Chapter 4 and why it is appropriate for distributed training.

2.2.2 Distributed Anomaly Detection

With the growth in computing power of edge devices, there is a potential to perform computation closer to the application [115]. Amongst the many challenges Edge AI faces, one is the case in which the data is distributed across different sites. This section reviews the literature of anomaly detection models that are built without transmitting the raw data.

Miao *et al.* [118] reformulates the One Class Support Vector Machine (OCSVM) into a decentralised optimisation function and diffusion cooperation [84] is used to achieve consensus. Similarly, O'Reilly *et al.* [147] reformulates the Minimum Volume Elliptical Principal Component Anal-

ysis (MVE-PCA) equations such that each site manages its own objective and constraint term. Then Alternating Direction Method of Multipliers [2] is used to determine the principal components through convergence via the mean of the objective values. Tsou *et al.* [186] proposes an ensemble algorithm to optimally weight Random Forest (OWRF) models trained on data at each site. The authors minimise uncertainty of predictions to learn the best model. Lyu *et al.* [112] merges Gaussian clusters trained on data from different levels of the network to the cloud. In the above works, only the summary of the data or parameters necessary for model training are being transmitted.

By analyzing the topology and installing rules to collect statistics at optimal monitoring positions, forwarding anomalies are detected and located in the network [103]. For a prediction variance anomaly detector, the most vital component is the covariance matrix. Instead of collecting all data segments centrally, the matrix is aggregated using compressed difference sequences and the sample standard deviation at each site for anomaly detection in WSNs [205].

Weights of Gated Recurrent Units trained at different sites are averaged based on number of data points for anomaly detection [137]. Federated learning (FL) is used to train a deep Long Short Term Memory model for anomaly detection with a focus on communication efficiency through a gradient compression mechanism [109]. The general principle of FL is to share parameters or gradient updates to train neural networks instead of data and is further explained in Section 2.2.3. These models are employed only on one dimensional sequential data. A Multi-task Deep Neural Network is trained using FL in a supervised manner for anomaly detection and classification in network data [216].

However, these methods involve *ad hoc* averaging of parameters which may not be identifiable because trained parameters on different datasets may represent data from different distributions; or these methods require the assumption that the objective function is linearly separable, which is

not necessarily valid. By *ad hoc* averaging, it is meant that these methods do not account for uncertainty or variance of the distribution of the data or that the data could be from a completely different distribution. Other methods of aggregation are problem specific and work simply on one-dimensional variables. Training deep learning models at the edge is infeasible, both computationally and in terms of memory. The scenario where the data is biased and unevenly distributed is not experimentally evaluated in the above studies.

2.2.3 Distributed Training Methods

This section considers existing methods for training a model in a distributed fashion. The consensus algorithm [160, 168, 203] is a well known method used to achieve agreement between sites. It is used in blockchain technology [98]. Liu *et al.* [107] perform neural network training using the consensus algorithm. Chahal *et al.* [29] surveys various methods such as synchronous and asynchronous stochastic gradient descent, gradient accumulation, scatter-reduce-all gather method, binary blocks algorithm and fault tolerance-all reduce method for neural network training.

Hardy *et al.* [73] achieves distributed training of a Generative Adversial Network (GAN) using a parameter server and a peer-to-peer communication pattern between the GAN's discriminator and generator. Teerapittayanon *et al.* [183] proposes a hierarchical method to train deep neural networks. Similarly, data communication is improved for distributed synchronous Stochastic Gradient Descent (SGD) by merging gradients of different layers [169]. The trade off between local gradient descent updates and global aggregation is explored with respect to resource constraints to optimise communication in [194]. Optimising SGD algorithms for communications only leads to slight a improvement and it depends on the task and data. If the data is unevenly partitioned the gradient updates can completely negate each other.

Lyu *et al.* [111] presents a privacy-preserving deep learning framework in IoT using random projection and differential privacy. Scardapane *et al.* uses decentralized Average Consensus method (DAC) and Alternating Direction Method of Multipliers to train a non-Bayesian RVFL [163].

These methods either have high computational complexity or have long training times and are not easily implemented at the edge. They are also not applicable for Bayesian approaches as they do not capture the uncertainty in the data at different locations as well. Again they are not evaluated with the scenario where the data is biased and unevenly distributed.

FL is a concept introduced for training models with decentralized data [117]. The key challenges of FL are that the training data are not Independent and Identically Distributed (IID), are unbalanced and massively distributed, and communication is also limited [104, 117]. There are studies which provide frameworks for FL [97] and tackle accuracy degradation in models due to imbalances in distributed training data [48]. A model similarity scheme to compare models trained on data in different locations and a protocol to classify data without transmitting model parameters helps preserve privacy but does not elaborate on distributed training [86].

As will be shown in Chapter 4, the proposed method implicitly achieves FL with certain restrictions such as the use of a SLFN. Communication is reduced since convergence is achieved quickly. This work is not a general FL method but shows that for specific applications such as anomaly detection, FL can be achieved with the right model design and methodology.

There are few studies which perform distributed training using Bayesian methods. Neiswanger *et al.* [134] performs parallel MCMC to obtain subposterior in each site using a fractional prior and then combining them. But the fractional prior may be too weak to effectively regularize [65]. Another approach would be to combine the posteriors of each site and divide the result by $K - 1$ priors, where K is the number of sites. This however, runs into computational instabilities [65]. Hasenclever *et al.* [74] develop

Stochastic Natural Gradient EP, a double loop optimization of power EP [119] to ensure convergence with additional computational complexities [190]. Xu *et al.* [211] also use EP and MCMC to perform inference at each site using a posterior server which maintains a global approximation. A deep BNN is more computationally demanding as it requires multiple stochastic passes [124]. For the purpose of anomaly detection within an application, this thesis shows that a SLFN produces near similar results with a huge reduction in computation and time.

2.3 Datasets and Evaluation Metrics

2.3.1 Sequential Datasets

As this thesis considers dynamic scenarios which occur mostly in streaming data, sequential or streaming datasets are required to validate the proposed framework. There are few real world datasets of such available [62, 122]. The streaming nature is important because similar data to training data will show lower variation in the Reconstruction Errors (RE)s of each batch, compared to unseen data as will be shown in Chapter 3.

The UNSW-NB15 dataset has a hybrid of real and modern, normal and contemporary synthesized attack [122, 123]. It has a volume of 100 GB with 2,540,044 network flow records which are logged in four CSV files. These flow records aggregate transmitted packets over the network based on source and destination IP address, port numbers and protocol of the connection into statistical features. The UNSW-NB15 training and test set CSV files summarise the different types of traffic in the entire dataset. This is a good dataset to test machine learning (ML) algorithms because the number of anomalies and normal data points are well balanced. In a real scenario, there will usually be more normal data than anomalous. This is also clear in the ground truth CSV file where the attacks are time stamped. Fortunately, the dataset comes with the sequentially recorded

network flow records.

Another network flow dataset is the CTU13 botnet dataset [62], which consists of 13 different captures of different real world botnets mixed with normal and background data, which is also provided sequentially. More details on these datasets are found in Appendix A.

2.3.2 Other ML Datasets

The NSL-KDD 2009 dataset is an improved version of KDDCUP dataset captured by 1998 DARPA IDS evaluation program [43, 90]. Though the NSL-KDD 2009 dataset is old, it is still a viable public dataset as it is still being used in many studies. It contains four different types of attacks namely, DoS (Denial of Service), Probe, R2L (Remote to Local) and U2R (User to Root). More details can be found in the appendix A.

The UCI machine learning repository contains many datasets for ML [47]. For anomaly detection, the class with more data samples is considered normal class and the others are combined to form the anomaly class in the used datasets in this thesis. Four datasets from different applications are used, namely, Abalone, PageBlocks, Shuttle and Australian Credit Approval. The datasets are randomly partitioned to achieve a balanced set for testing in Chapter 4. The details are found in Table 2.2.

Table 2.2: Datasets Summary

Dataset	**Application Domain**	**Attributes**	**No. Instances Training (Testing) (Normal+Anomaly)**
UNSW-NB15	Intrusion Detection	42	56000 (37000+45332)
NSLKDD2009	Intrusion Detection	41	67340 (9711+12833)
Abalone	Biology	8	1880 (94+94)
PageBlocks	Text Recognition	10	4353 (560+560)
Shuttle	Sensor Monitoring	9	34108 (3022+3022)
Australian Credit Approval	Finance	14	283 (100+100)

2.3.3 Evaluation Metrics

The following measures are used to evaluate the methods developed in this thesis: Receiver Operating Characteristic (ROC) curve, Area Under ROC Curve (AUC), Precision-Recall (PR) curve, True Positive Rate (TPR), False Positive Rate (FPR) and Accuracy (Acc).

The ROC curve is the plot of TPR against FPR at different threshold settings. The closer the curve is to the left and the top border of the ROC space, the more accurate the model, as illustrated in Figure 2.4. The AUC is a scalar measure of the ROC curve, and represents the degree of separability of anomalies versus normal data. A perfect model has an AUC value of 1 while a value of 0.5 or less suggests that the model has no capacity to separate the classes.

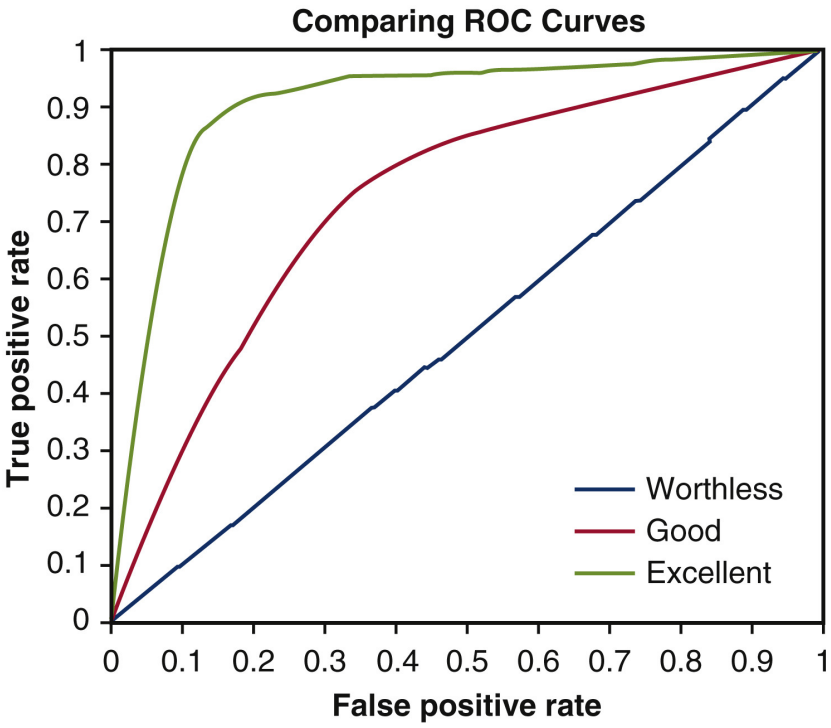


Figure 2.4: Receiver Operating Characteristic (ROC) curve

For streaming data, when the number of anomalies are fewer than the

normal data, the methods are also evaluated on the Precision-Recall (PR) curve. The PR curve is the plot of precision against Recall (also known as TPR) as shown in Figure 2.5. The area under PR curve gives the Average Precision (AP). The baseline of the PR curve is the proportion of anomalies in the dataset.

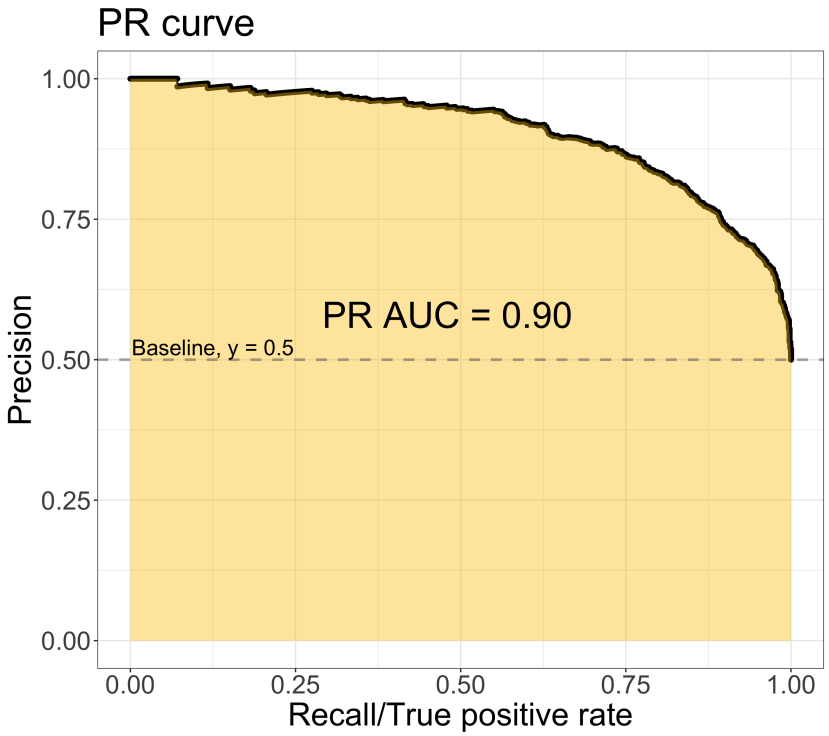


Figure 2.5: Precision Recall (PR) curve

TPR is also known as detection rate. True positives (TP) are the number of actual anomalies detected as anomalies. False positives (FP) is the number of normal data detected as anomalies. True negatives (TN) are number of normal data detected as normal. False Negatives (FN) are the number of anomalies detected as normal.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.3}$$

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{Total number of data points}} \tag{2.4}$$

To evaluate convergence of a BNN with covariance matrix $\mathbf{Q}$ and mean $\mathbf{r}$, two other measures are introduced to measure relative difference to global parameters [147]. The details are explained in Chapter 4. Let $\mathbf{r}_G$ and $\mathbf{Q}_G$ denote the global parameters and $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm. For $\mathbf{r}_G$, the relative difference of the mean square error along the independent dimensions is computed:

$$E_{rel}(\mathbf{Q}) = \frac{\|\mathbf{Q} - \mathbf{Q}_G\|_{\mathcal{F}}}{\|\mathbf{Q}_G\|_{\mathcal{F}}} \tag{2.5}$$

$$E_{rel}(\mathbf{r}) = \frac{\frac{1}{d}\sum_{j=1}^{d} \|\mathbf{r}_j - \mathbf{r}_{Gj}\|_2}{\frac{1}{d}\sum_{j=1}^{d} \|\mathbf{r}_{Gj}\|_2} \tag{2.6}$$

To evaluate the real world scenarios however, since the data is not labelled, it is difficult to obtain a quantitative measure of performance. Thus, in relation to the scenario and with feedback from the experts, the methods are evaluated on whether the models detect the anomalies based on a threshold. This threshold is determined based on the model's performance on normal test data.

2.4 Conclusion

Anomaly detection literature is rich and many alternative techniques and non machine learning methods such as Artificial Immune Systems can also found [42,77]. The appropriateness of these methods depend on the application domain and whether the underlying assumptions are met. Machine Learning methods are appropriate when there are many features to consider and when patterns are not easily recognisable.

This chapter has reviewed the state of the art in anomaly detection methods. The methods can be classified as Online or Offline, and each

of them has their own strengths and demerits. In this review, deep learning methods in particular have been shown to perform well for feature extraction and anomaly detection.

Detecting anomalies under evolving data is a gap in the current research which needs to be addressed. Offline methods attempt to generalise to new data but there is no way of verifying whether the method has generalised properly. It is also prudent to detect new normal data and understand them. Online methods handle concept drift but suffer from lack of labelled training data for accurate detection. Moreover, an Online Offline combination method for behaviour based anomaly detection is lacking.

Then a review on various methods to train a Bayesian Neural Network was given. Distributed models and methods to train a distributed model were surveyed. However, these methods are not validated on heterogeneity of data at different locations. Moreover, the methods involve *ad hoc* averaging of parameters or require the assumption that the objective function is linearly separable, which is not necessarily valid. Existing Bayesian methods have high computational complexity for edge deployment. There is also no fully Bayesian approach to distributed training for anomaly detection.

Hence, these aspects form the motivation for the approaches proposed in the following chapters. The methods developed are also validated on real world domains. The next chapter addresses how anomaly detection can take into consideration evolving data.

Chapter 3

The Hybrid Online Offline Framework

To address the issue of detecting anomalies under evolving data, this chapter develops a novel Hybrid Online Offline Framework and it is shown to improve on any individual online or offline model. The main aim is to exploit strengths of the individual models while avoiding their weaknesses [141–144]. Deep learning models, trained offline, are able to pick out patterns in high-dimensional data, find succinct representations and learn the underlying distribution of the data, but they cannot be updated easily. Online models are better able to spot anomalies (outliers) in new data, but do not possess the knowledge of the distribution of normal data. Hence, the rationale for the hybrid is to use the offline model's learned knowledge as a bias for the online model to select data to train on. The main framework and algorithm is explained along with the assumptions required and the roles of each component.

Following the framework, a heuristic method to detect new data is presented in Section 3.3. Instead of generalising to new data, the framework deems all new data as anomalous. Then through clustering, new potentially legitimate data is detected. This also prevents generalising incorrectly to new data. This thesis is the first to consider such an approach.

In Section 3.4, a signature-based variant of the hybrid framework is explored.

3.1 The Online Offline Framework

The overall framework is shown in Figure 3.1 and the main method is found in Algorithm 1 [142,144]. Table 3.1 lists the symbols used.

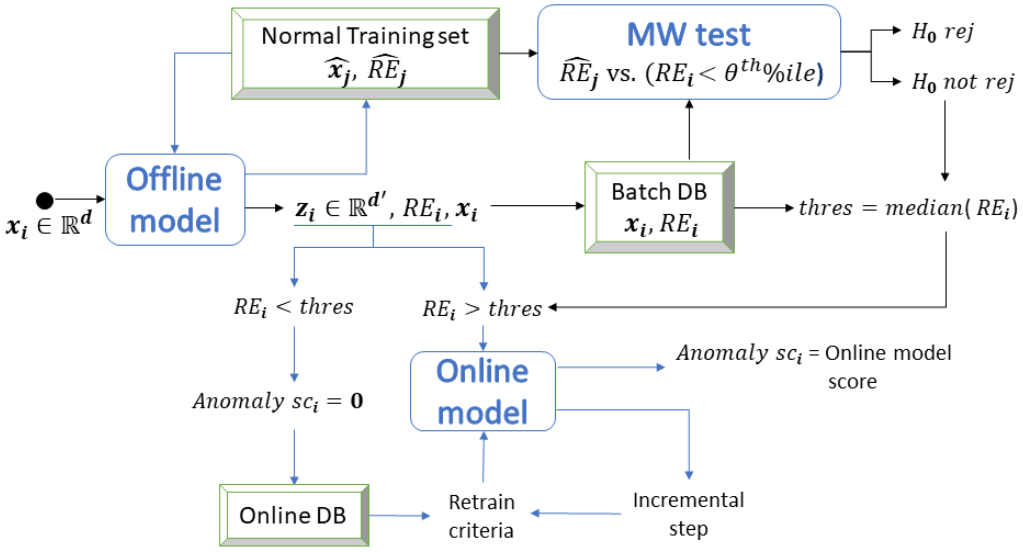


Figure 3.1: Online Offline Framework

There are many components to the framework. In the beginning, the offline model is trained with normal data $\{\widehat{\mathbf{x}}_j\}_{j=1}^{n}$ and it outputs the Reconstruction Error $\widehat{RE_j}$ for each point $\widehat{\mathbf{x}}_j$ (lines 1-2, Algorithm 1). This offline model can be any deep learning model that outputs a form of anomaly score such as RE and provides dimensionality reduction. In the experiments in Section 3.2, three different offline deep learning models are evaluated. One key difference from other methods is that this model learns the existing normal data or the training dataset well and trades bias for higher variance, which prevents generalising incorrectly on new data as will be

Table 3.1: Description of symbols

Symbol	Description
$\widehat{\mathbf{x}}_j$	Normal data used for offline training, $j = 1, ..n$
$\widehat{\mathbf{x}}_k$	Normal data used for online initialisation, $k = 1, ..m$
$\mathbf{x}_i$	Incoming live stream data to be scored
d	Dimension of training and incoming data
d'	Dimension of latent layer representation, $d' < d$
$\mathbf{z}_i$	Latent layer representation of $\mathbf{x}_i$
$\widehat{RE_j}$	Reconstruction error of $\widehat{\mathbf{x}}_j$ through offline model
RE_i	Reconstruction error of $\mathbf{x}_i$ through offline model
$Anomaly\ sc_i$	Anomaly score for $\mathbf{x}_i$
θ	Upper percentile bound on the batch, i.e. Remove outlier points $> \theta$-percentile
$thres$	Threshold to determine data for online training
n	Number of data used for offline training, $n > m$
m	Number of data in each batch to determine threshold
l	Number of data in each batch for online training
Offline model	A type of Autoencoder or any model that provides RE
Online model	An outlier detection model
Online DB	Online database to store data for online training
Batch DB	Batch database to store data for MW test
MW test	1-sided Mann-Whitney U test
H_0	Null hypothesis: Mean ranks equal or populations same
H_a	1-sided alternative hypothesis. Batch mean ranks are larger than those of legitimate data
Temp-DB	Temporary search database to store data for clustering
M	Number of data points to perform clustering on
t	Split range into t equal intervals

Algorithm 1 Online Offline Framework

OfflineModel ← Train($(\widehat{\mathbf{x}}_j)_{j=1,..n}$, parameters)
$(\widehat{RE_j})_{j=1,..n}$ ← *OfflineModel*($(\widehat{\mathbf{x}}_j)_{j=1,..n}$)

Initialize *OnlineModel* ← Train($(\widehat{\mathbf{x}}_k)_{k=1,..m}$, parameters)
$thres$ ← Median($(\widehat{RE_k})_{k=1,..m}$)

while True **do** ▷ Stream data $\mathbf{x}_i$
  $\mathbf{z}_i$, RE_i ← *OfflineModel*($\mathbf{x}_i$)
  if $RE_i < thres$ **then**
    $Anomaly\ sc_i \leftarrow 0$
    Online DB ← $\mathbf{z}_i$
  else
    $Anomaly\ sc_i$ ← *OnlineModel*($\mathbf{z}_i$)
  end if
  Batch DB ← $(\mathbf{x}_i, RE_i)$
  if Retrain Criteria **then** ▷ Can be parallelized
    OnlineModel ← Update(Online DB, *OnlineModel*)
    Online DB ← Empty(Online DB)
  end if
  if Size of Batch DB == m **then** ▷ Can be parallelized
    BatchData ← RemoveOutlier($(RE_i)_{i=1,..m}$, θ)
    p-value← *MW test*($(\widehat{RE_j})_{j=1,..n}$, BatchData)
    if p-value $> 5\%$ **then**
      $thres$ ← Median($(RE_i)_{i=1,..m}$)
    end if
    Batch DB ← Empty(Batch DB)
  end if
end while

shown in Section 3.3.

The online model is also initialised with existing data. The threshold $thres$ is initialised as the median value of the dataset used to initialise the online model (lines 3-4, Algorithm 1). The online model is an outlier detection algorithm. The parameters can be updated with an incremental step and with new data or completely retrained with new data, i.e. 0% knowledge retention rate. The framework is evaluated with six different outlier detection methods of which two of them are incremental.

As the stream of values $\mathbf{x}_i$ arrives, it is fed into the offline model to obtain the low-dimensional representation $\mathbf{z}_i$ and RE_i (line 6). If $RE_i < thres$, it is classified as normal, assigned anomaly score of 0, and stored in an online Database. The online model is retrained with this new data once a retraining criterion is met. If $RE_i \geq thres$, the current online model provides the anomaly score for $\mathbf{x}_i$ (line 11). The stream of $\mathbf{x}_i$s and its corresponding RE_is are then collated into batches (Batch DB) to obtain $(RE_i)_{i=1,..m}$ (line 13, 19). RE_i less than θ^{th} percentile of the batch (Remove-Outlier function in line 19) is used to conduct a one-sided Mann-Whitney test (see 3.1.3) against $(\widehat{RE}_j)_{j=1,..n}$ (line 20). If the null hypothesis H_0 is not rejected, the median value of this batch will be the new threshold $thres$. This $thres$ is allowed to drift. If H_0 is rejected, the $thres$ value is not updated because more than θ% of the data is significantly different from the normal training set $(\widehat{\mathbf{x}}_j)_{j=1,..n}$. In the following subsections, the functions of each step shall be explained.

3.1.1 Offline Model

The model of choice is a deep AE for the reasons mentioned in chapter 2 and that deep learning has shown to be able to learn complex data structures such as the underlying distribution of high dimensional data [126].

The REs of the stream data are compared against a threshold for the anomaly score which is either 0 or determined by the online model. If this

threshold was fixed, then it is comparing solely with the offline training dataset. In which case, if the data point has an anomaly score of 0, this data point goes into training the online model and the online model will be learning the same thing as the offline model. Thus, a moving threshold is required to allow data that are not similar to the normal training data set for online training. With the drifting threshold, the aim is to include new variants of normal data in the stream. This is prevalent when data types are evolving as these variants would have higher REs.

When the collective REs of a batch of data decreases, lowering the threshold helps to detect anomalies better and improve detection rate. When it increases, raising the threshold will prevent normal data from being classified incorrectly or reduce false positives. Section 3.1.2 describes the moving threshold in detail.

3.1.2 Online Model

The online model is trained in batches with recent data, and has a lower training complexity compared to the offline model. In deployment, supervision is not easy because the streaming data is unlabelled. Without any labels, it performs outlier detection purely under the assumption of that the normal data occurs in relatively large numbers. Thus, to help it to perform better, it can be trained with data points that are closer to known normal data profiles. The offline model provides such supervision (line 7,9, Algorithm 1). In addition, the online model can be trained using the latent layer representation $\mathbf{z}_i$, addressing the curse of dimensionality.

There are many outlier detection models that can be used as the online model as described in section 2.1.3. This thesis explores the OCSVM, GMM, LOF and KDE. The ideal online model would also be able to retain knowledge or incrementally learn with new data and two of such methods, Incremental OCSVM (IOCSVM) and Incremental GMM (IGMM) are expolored.

The OCSVM can retain knowledge in the form of its "support vectors", which describe the current center and decision boundary completely. The method of retaining all of the support vectors (100% knowledge retention) and using new data to update the model shall be denoted as IOCSVM. The scenario where no support vectors are retained (0% knowledge retention) shall be denoted as the traditional OCSVM in this thesis. There are other works applying OCSVM to the latent layer of an AE [27, 51, 195], but no methods have considered the streaming nature of the problem and implemented it in an incremental fashion.

The IGMM updates parameters through merging similar components and regularising the weights to sum to one [1]. The implementation of IGMM in [1] uses forgetting rate equivalent to $1-$(knowledge retention rate) in this thesis. The model retrained completely with new data (0% knowledge retention) shall be denoted as the GMM.

The existing incremental versions of LOF [125, 151] perform updates one point at a time, which is infeasible in live network streams. Re-training with data from current and previous iterations will incur high memory overhead. KDE with the Gaussian kernel fits a Gaussian at each point while GMM fits over the number of components. Hence, the incremental version of KDE [172] can be considered as similar to IGMM.

Though the threshold drifts, it needs to drift with structure and independently from the normal training set. Here is where an assumption needs to be invoked. If a certain percentage of the data in each batch is expected to be normal, the percentile value can be used in determining a threshold. In the experiments, 50% of the data in each batch of the stream is assumed to be normal. This is a conservative assumption in real-scenarios, thus, the median value of the REs of the batch is taken as the threshold (line 22, Algorithm 1). Percentile values are robust to outliers. To reiterate, there may be normal data above this threshold, but they are not considered for online training. They can still be given a low anomaly score by the online model, albeit not 0 (line 11, Algorithm 1).

However, allowing this threshold to drift freely without any restriction might result in anomalous data going into online training. In situations such as DoS attacks, when there are more anomalies in the batch, the median value will increase. To prevent this, the REs of each batch is compared with the normal training dataset using the Mann-Whitney U statistical test which will be described in section 3.1.3.

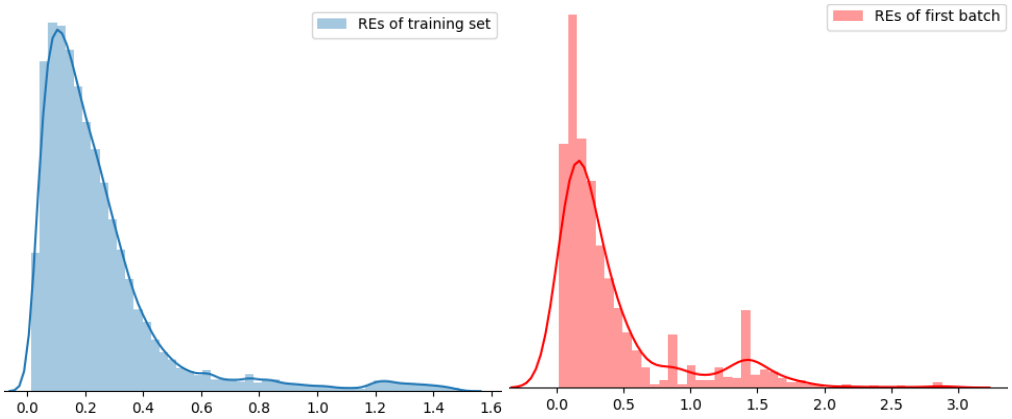
3.1.3 Mann-Whitney U (MW) Test

The REs of points in each batch form a distribution. To determine whether the profile of the data has substantially changed, the distribution of REs of each batch is compared with the REs of the training data. This is to prevent updating it when there are anomalous data in the stream.

The Mann-Whitney U test (MW-test) also known as the Mann-Whitney-Wilcoxon or the Wilcoxon rank-sum test is a non parametric test to determine whether the medians of two independent groups are significantly different [56, 114]. It can also be interpreted as testing whether observations from one group tend to be larger than observations in the other. In depth formulation of the MW-test is found in [171].

To perform this test, the following assumptions need to hold. Firstly the data must be ordinal, i.e. they can be ranked. Secondly, the two groups of data must be independent. Thirdly, each observation must be mutually independent. Fourthly, the two distributions should have the same shape. This last assumption is not a hard requirement [56]. If all four assumptions are met, the MW-test can be used to determine the exact difference between the medians of the two groups. If only the first three assumptions are met, the MW-test determines whether one group's mean ranks are significantly higher or lower. The REs are continuous values which can be ranked from lowest to highest and each new batch of data is independent. There is no relationship between each data point considered in the datasets used. Under the assumption that most data is normal, a right

skewed distribution is expected as shown in Figure 3.2.



(a) Training set (b) First batch containing anomalies

Figure 3.2: Distribution of reconstruction error on UNSW-NB15

A one-sided MW-test is performed to check whether each batch's REs are significantly greater than that of the training set. When this test is statistically significant (when the p-value is small and H_0 is rejected), the batch contains more points further from the distribution of the training data. In other words, more anomalies or new types of data that are different from the ones seen during training are present. In this scenario, the threshold is not updated with the new median value obtained from this batch.

When there are anomalies in the stream, this test is naturally going to be statistically significant. Excluding high REs, which denote anomalous points allows us to compare normal data against each other, (line 19, Algorithm 1). Hence, this test is better performed with only the data points in the batch that are less than a fixed θ^{th}-percentile value of the batch, (line 20, Algorithm 1). $100 - \theta$ can also be seen as the percentage of expected anomalies in each batch. A conservative value will be the $\theta = 50$, which is similar to the threshold for selecting data for online training. If the test is statistically significant at $\theta = 50$, this means that more than 50% of the data

in that batch is different from the training data and could point towards a scenario such as an intrusion attempt, network fault or DoS in network applications. However, this will not occur all the time. To capture new variants of normal data that the offline model has not seen, $\theta > 50$ is more appropriate.

3.2 Results and Analysis

To evaluate the Hybrid Online Offline Framework, sequential datasets are used as described in Secion 2.3.1. Normal data from the first 100,000 flows of the first CSV file from the UNSW-NB15 dataset are selected as the training set. This resembles a scenario in which not all types of normal data are available during the training phase. There are 88,570 normal flow records in the first 100,000 records, and 10% is held as a validation set for training of the AE. Normal data from the next 1000 records are used to initialise the online model. Anomaly detection is performed on the remaining 119,000 flow records. All the anomalies in the first CSV file are contained within these records. The model's performance is also evaluated on batches with no anomalies as well.

To test the model to be able to perform under new normal data types in Section 3.3, two types of data are removed within the dataset from training. *ftp* and *ssh* from the "*services*" feature are removed from the training dataset and online initialisation while including them in the testing set. Then Algorithm 2 is utilised to detect these data types later. This simulates the scenario of encountering completely new data. *ftp* and *ssh* were chosen because they present a substantial but not significant portion of the dataset in terms of both normal data and anomalies.

From the CTU13 botnet, four different captures from scenarios which contain a large number of records are selected, namely CTU13-13, -9, -1 and -3. From each 200,000 chronological records containing anomalies are selected. The training set is obtained in the similar fashion as the UNSW-

NB15 dataset such that the number of training items are near similar. The remaining data is used as the testing set.

Details on the preprocessing of the data, model parameters and time taken for training are described in appendix B.1.1.

3.2.1 Hybrids vs. Pure

The results are summarised in Table 3.2. The models based on the proposed framework is denoted as "Hybrids", also individually denoted as AE+MW, VAE+MW, DAE+MW (+ the respective online model). The purely online models (which perform only outlier detection in batches on the latent layer representation of the respective offline models) are denoted as online AE, VAE, DAE + the respective outlier detection model (and the offline versions similarly). In this scenario, neither MW test nor thresholds are used to select data for online training. The vanilla AE, VAE and DAE is also considered as a purely offline model using RE as the anomaly score. UNSW-NB15 refers to the complete dataset, while UNSW-NB15-New refers to the scenario where *ftp* and *ssh* were removed from the training set but included in the test set. This is to show-case the detection of new data later in Section 3.3.

The best model overall is the hybrid AE+MW+IOCSVM with an average AUC of 0.9372. Figures 3.3 and 3.4 show ROC curves for each methods using AE as the offline model, on UNSW-NB15 and UNSW-NB15-New respectively. Figures 3.3a and 3.4a compare the hybrid models. The AE+MW+IOCSVM is also included in the comparison between the individual online and offline models. Similarly, Figures 3.5 and 3.6 give the PR curves for each of the AE methods run on UNSW-NB15 and UNSW-NB15-New. In the legend of Figures 3.5 and 3.6, the average precision or the area under PR curve is reported for each of the model.

The hybrids show the best performance on average in terms of AUC over individual online and offline models regardless of choice of AE, VAE

	Models		UNSW	UNSW	CTU13	CTU13	CTU13	CTU13	Row	Model
	Offline	**Online**	-NB15	-NB15-New	-13	-9	-1	-3	**Average**	**Average**
Hybrids	AE + MW	IOCSVM	0.9506	0.8821	0.9253	0.9513	0.9674	0.9466	**0.9372**	**0.8769**
		IGMM	0.9144	0.8845	0.9743	0.8308	0.9672	0.8178	0.8981	
		OCSVM	0.9201	0.8680	0.8459	0.8591	0.6151	0.8767	0.8308	
		GMM	0.9158	0.9274	0.8349	0.8228	0.8521	0.8723	0.8708	
		LOF	0.8982	0.9016	0.8168	0.8428	0.9505	0.8887	0.8831	
		KDE	0.9529	0.8864	0.8458	0.8205	0.6804	0.8626	0.8414	
Online	AE[1]	IOCSVM	0.7305	0.6129	0.6222	0.5423	0.8324	0.5753	0.6526	0.6644
		IGMM	0.4087	0.5073	0.8523	0.2685	0.6330	0.8887	0.5930	
		OCSVM	0.9056	0.7265	0.9365	0.4864	0.4698	0.8897	0.7357	
		GMM	0.6132	0.5889	0.8589	0.3267	0.6972	0.8742	0.6598	
		LOF	0.5818	0.6084	0.4974	0.5140	0.6661	0.4062	0.5456	
		KDE	0.9733	0.8433	0.8575	0.6752	0.5700	0.8806	0.7999	
Offline	AE		0.8978	0.8484	0.8723	0.8407	0.9175	0.8654	0.8736	0.8109
	AE[1] + OCSVM		0.9215	0.8502	0.4728	0.6141	0.8734	0.7617	0.7489	
	AE[1] + GMM		0.8858	0.8898	0.9124	0.9477	0.7117	0.8498	0.8662	
	AE[1] + LOF		0.8428	0.8164	0.6240	0.7125	0.9385	0.3257	0.7099	
	AE[1] + KDE		0.9850	0.8774	0.8630	0.8456	0.6597	0.9038	0.8557	
Hybrids	VAE + MW	IOCSVM	0.7938	0.9187	0.9285	0.5947	0.7174	0.9295	0.8137	**0.8060**
		IGMM	0.8438	0.8871	0.9321	0.5662	0.9590	0.9339	0.8536	
		OCSVM	0.8331	0.8863	0.9220	0.5815	0.6819	0.8946	0.7999	
		GMM	0.8385	0.8799	0.8375	0.5643	0.8293	0.8737	0.8038	
		LOF	0.8332	0.8923	0.8728	0.5667	0.8699	0.7708	0.8009	
		KDE	0.8715	0.8719	0.8615	0.6267	0.5923	0.7623	0.7643	
Online	VAE[1]	IOCSVM	0.4998	0.3704	0.5489	0.4832	0.6037	0.4977	0.5006	0.6333
		IGMM	0.4153	0.6029	0.7195	0.4893	0.6697	0.8631	0.6266	
		OCSVM	0.7626	0.7787	0.8781	0.6187	0.3293	0.8902	0.7096	
		GMM	0.5476	0.6219	0.7974	0.6216	0.7060	0.8328	0.6878	
		LOF	0.6386	0.6020	0.5449	0.5242	0.6954	0.1964	0.5335	
		KDE	0.8560	0.8503	0.8029	0.8616	0.2518	0.8284	0.7418	
Offline	VAE		0.7355	**0.9345**	0.8694	0.6487	0.8530	0.8889	0.8216	0.7846
	VAE[1] + OCSVM		0.9388	0.8592	0.7533	0.2067	0.4036	0.4773	0.6064	
	VAE[1] + GMM		0.8828	0.8973	0.8849	0.9591	0.9389	0.8978	0.9101	
	VAE[1] + LOF		0.8587	0.8248	0.8073	0.7176	0.8836	0.8464	0.8230	
	VAE[1] + KDE		0.9246	0.8750	0.8129	0.9247	0.1707	0.8635	0.7619	
Hybrids	DAE + MW	IOCSVM	0.9209	0.8639	0.9472	**0.7954**	0.9082	0.7582	0.8656	**0.8652**
		IGMM	0.9212	0.8293	0.9161	0.9583	0.9847	0.9190	0.9214	
		OCSVM	0.8940	0.8333	0.9094	0.7733	0.8405	0.8672	0.8529	
		GMM	0.8926	0.8619	0.8756	0.8880	0.9078	0.8771	0.8838	
		LOF	0.8313	0.8597	0.9053	0.9011	0.9700	0.9063	0.8956	
		KDE	0.9475	0.7767	0.8207	0.6722	0.7015	0.7127	0.7718	
Online	DAE[1]	IOCSVM	0.7192	0.7087	0.6295	0.4209	0.7991	0.4999	0.6295	0.6414
		IGMM	0.4524	0.5487	0.7339	0.5689	0.5107	0.7828	0.5995	
		OCSVM	0.8819	0.6415	0.8458	0.4403	0.5866	0.8449	0.7068	
		GMM	0.5650	0.6287	0.7739	0.4833	0.6394	0.8098	0.6500	
		LOF	0.6184	0.6315	0.4819	0.5150	0.6942	0.4630	0.5673	
		KDE	0.9672	0.6454	0.7803	0.4161	0.6472	0.7150	0.6952	
Offline	DAE		0.8816	0.8400	0.9596	**0.9677**	0.9215	0.8887	0.9098	0.7971
	DAE[1] + OCSVM		0.9050	0.7804	0.5891	0.3497	0.9114	0.2872	0.6371	
	DAE[1] + GMM		0.9153	0.8804	0.9215	0.9492	0.7912	0.8769	0.8890	
	DAE[1] + LOF		0.8345	0.8092	0.7053	0.7238	0.9338	0.5924	0.7665	
	DAE[1] + KDE		0.9866	0.8463	0.7957	0.5972	0.7296	0.7430	0.7830	

Table 3.2: Area under ROC curves (AUC) for the offline AE, VAE, DAE and online IOCSVM, IGMM, OCSVM, GMM, LOF, KDE. The hybrids denote the framework, the online denote purely online models and offline denote models whose training is done only once offline. [1] denotes that the model is used for dimensionality reduction only.

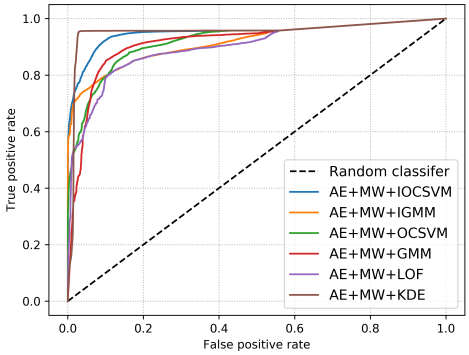
or DAE. The hybrid AE+MW+IOCSVM and hybrid AE+MW+IGMM performs better over the other hybrids because of its ability to retain knowledge. In the IOCSVM, the retention of the support vectors ensures that the decision boundary or the center of the OCSVM do not change drastically in each iteration. The purely online models do not perform as well, regardless of the offline choice (AE / VAE / DAE), because they have not seen the attributes of normal data. They do not differentiate between anomalies and infrequent but normal data types. Feeding them with appropriate inliers in the case of the hybrids helps them to score anomalies as outliers better.

Another interesting result is that sometimes the online OCSVM and the online GMM performs better than their incremental counterparts, the online IOCSVM and the online IGMM. This happens mostly in the purely online scenario. For example, with the AE, the online AE+OCSVM has a score of 0.9056 while the online AE+IOCSVM has a score of 0.7305. In the online IOCSVM an anomaly point with low RE can become a support vector. Retaining this point affects subsequent iterations. Likewise, components pertaining to certain anomalous regions gets retained in the online IGMM. The hybrid models are not severely affected by it because they have guidance to select the right points for training. One way to address this issue is to decrease the knowledge retention rate for incremental online methods.

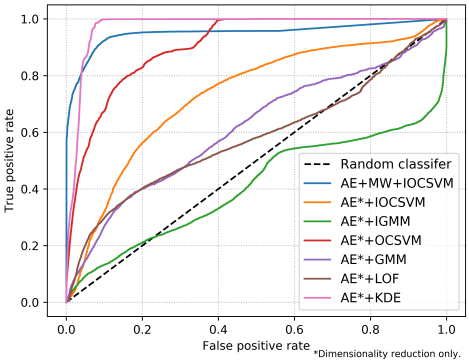
Performance of most models drop when going from the UNSW-NB15 dataset to UNSW-NB15-New. This can also be seen in Figure 3.6. The *ftp* and *ssh* data contains both normal and anomalous points. In the hybrid models, all of the *ftp* and *ssh* data have high REs because the AE has not seen this type of data before. Hence, all new data, normal or not, is classified as anomalous as expected. As mentioned in section 2.1.2, there is no model which can generalise well all the time in a real live network. Section 3.3 will show the importance of such a classification to detect all new data instead of generalising to new data. The ROC curves in Figures 3.3 and

3.4 also suggest that the models following the hybrid framework are more consistent compared to the others. Examining the PR curves, Figures 3.5 and 3.6 on the UNSW-NB15 dataset, all the models in the hybrid framework achieved an average precision over 0.5. Though ROC curves are always monotonic, PR curves need not necessarily be monotonic [41, 80]. In this aspect, the area under the PR curve or the average precision provides better comparability. The best performer is the AE+MW+IOCSVM with an average precision of 0.8472. On the UNSW-NB15-New dataset, though they outperform the random classifier, the average precision drops for all models. All of the new data, which contain normal and anomalies, are considered anomalous because the model has not seen it before. So this result is expected. The hybrid AE+MW+GMM has the highest average precision of 0.5706. It also has the highest AUC of 0.9274 amongst the models with AE as the offline model.

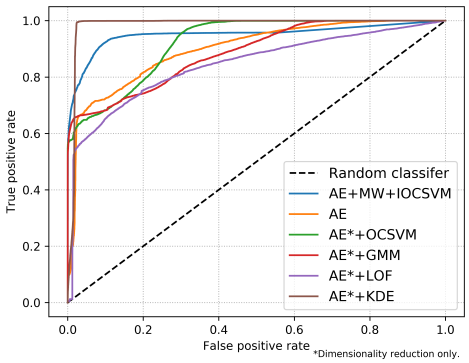
The vanilla DAE performs better on average with an average AUC of 0.9098 compared to the vanilla AE and VAE when comparing the offline models. However, when considering the hybrids, the AE outperforms the VAE and DAE. The VAE is a generative model and its RE is not particularly suited to identify inputs similar to the training distribution [127]. The VAE may also assign high probability (i.e. low RE) in areas where no data have been observed [173]. However, it is the best performing model with an AUC of 0.9345 under UNSW-NB15-New. This is because it generalises better to new data. The DAE is trained to remove noise from the data. Hence, it is not as well suited to select data for online training as the AE. The DAE also adds regularization into the training enabling the model to better generalise to new data. On contrary to common intuition, this is not a desirable feature to detect new data as explained in Section 3.2.4.



(a) Hybrid models

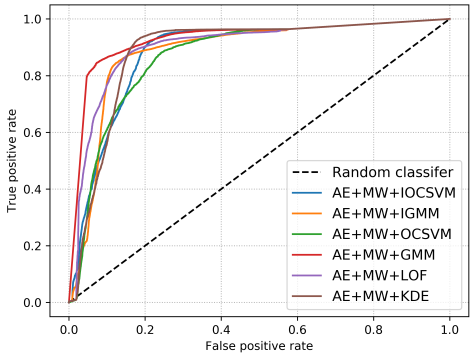


(b) Online models

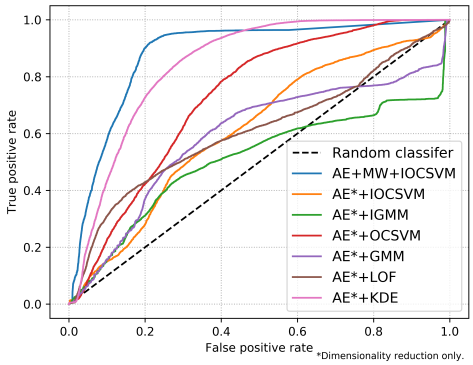


(c) Offline models

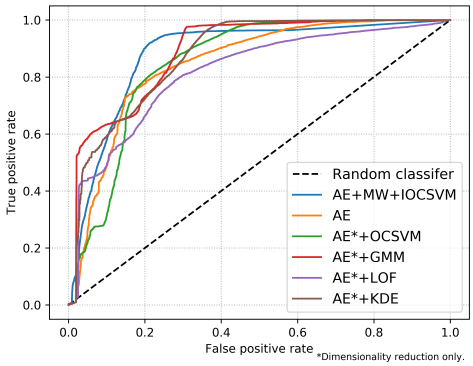
Figure 3.3: ROC curves on UNSW-NB15 with the AE. The hybrid models are the most consistent.



(a) Hybrid models

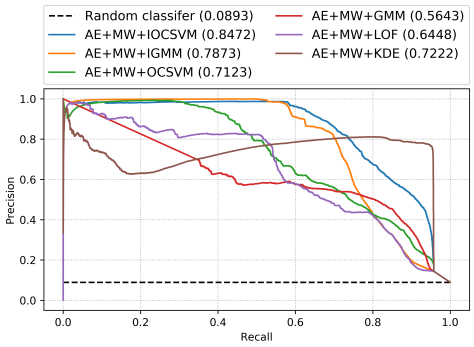


(b) Online models

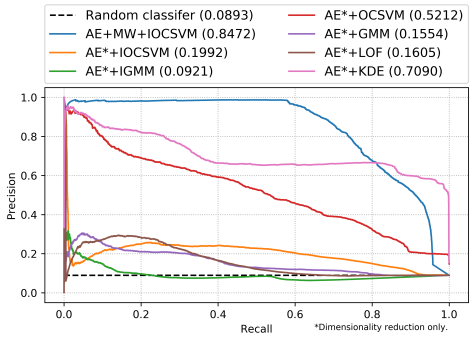


(c) Offline models

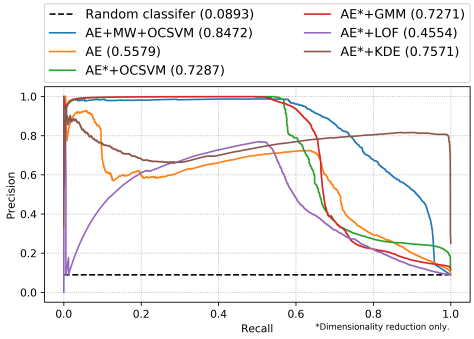
Figure 3.4: ROC curves on UNSW-NB15-New with the AE. The hybrid models are the most consistent.



(a) Hybrid models

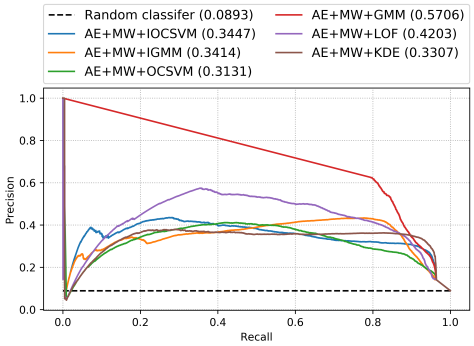


(b) Online models

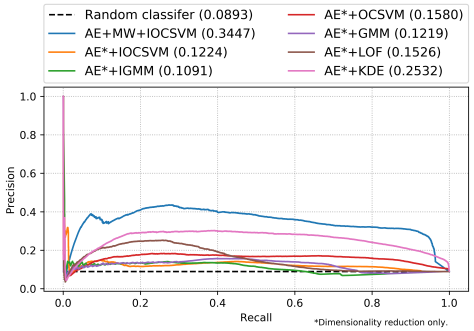


(c) Offline models

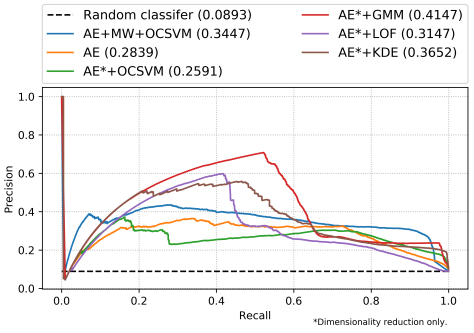
Figure 3.5: PR curves on UNSW-NB15 with the AE. The average precision is shown in the legend beside each label.



(a) Hybrid models



(b) Online models



(c) Offline models

Figure 3.6: PR curves on UNSW-NB15-New with the AE. The average precision is shown in the legend beside each label.

3.2.2 Detection Rates

To classify data as normal *vs* anomaly, a threshold is implemented on the anomaly scores. Unlike other models, the OCSVM also learns a decision boundary but this may not be optimal. The decision boundary can be affected if an anomaly point is selected for online training. The anomaly scores are unaffected by such points. They are related to the point's distance from the center which is more robust because it considers all of the points and the influence of each point is smaller. Table 3.3 gives results of using the decision boundary by AE+MW+IOCSVM *vs* an optimal threshold, on the anomaly scores on the UNSW-NB15 dataset. The "optimal" threshold here is the one for which detection rate equals overall accuracy. In the real world setting, this threshold can be determined by holding out a validation set of normal data and relevant anomalies. The anomaly scores give a better performance as compared to the decision boundary. However, this does come at a cost of higher false positive rate as shown in Table 3.4.

Table 3.3: Results for individual anomalies in the UNSW-NB15 of the AE+MW+IOCSVM

Anomalies	**Detection rate of decision boundary (%)**	**Detection rate of optimal threshold (%)**	**Total number present**
Fuzzers	24.85	61.87	1718
Exploits	61.59	93.94	1997
Generic	98.07	99.73	5599
Reconnaissance	39.63	90.31	805
DoS	47.85	91.42	303
Backdoors	58.97	92.31	39
Worms	50.00	50.00	8
Shellcode	29.79	84.04	94
Analysis	14.52	59.68	62

Table 3.4: Overall results of the AE+MW+IOCSVM

Method	FPR(%)	TPR(%)	Accuracy(%)
Decision Boundary	1.27	72.24	96.37
Optimal Threshold	8.86	91.13	91.14

3.2.3 Moving Medians

Table 3.5: Median thresholding on UNSW-NB15

Method: AE+MW+IOCSVM	**AUC**
Moving Median with MW	0.9506
Moving Median without MW	0.9262
Fixed Threshold	0.9467

Table 3.5 compares the AE+MW+IOCSVM method using the moving median threshold on the UNSW-NB15 dataset. Moving median with MW refers to the AE+MW+IOCSVM method. Moving median without MW refers to the method by allowing the threshold values to increase and decrease freely. The fixed median thresholding method fixes the threshold at median of the training set REs at initialisation.

Figure 3.7 shows median values of the REs of each batch on top, and the number of actual anomalies and number of predicted anomalies in each batch below. The line shows the number of anomalies predicted by the IOCSVM decision boundary. Red dots show batches in which the null hypothesis H_0 is rejected and the purple dots show the batches where it is not rejected. When it is *not* rejected, the threshold for selecting points for online training is allowed to drift and updated with the new median. When H_0 is rejected, this new median is discarded, suggesting more than $100 - \theta$ percentile of the batch contains anomalies or new data. The green line indicates the median of training REs.

In Table 3.5, using the MW test to accept or reject the new median threshold gives better performance than letting it freely move or using

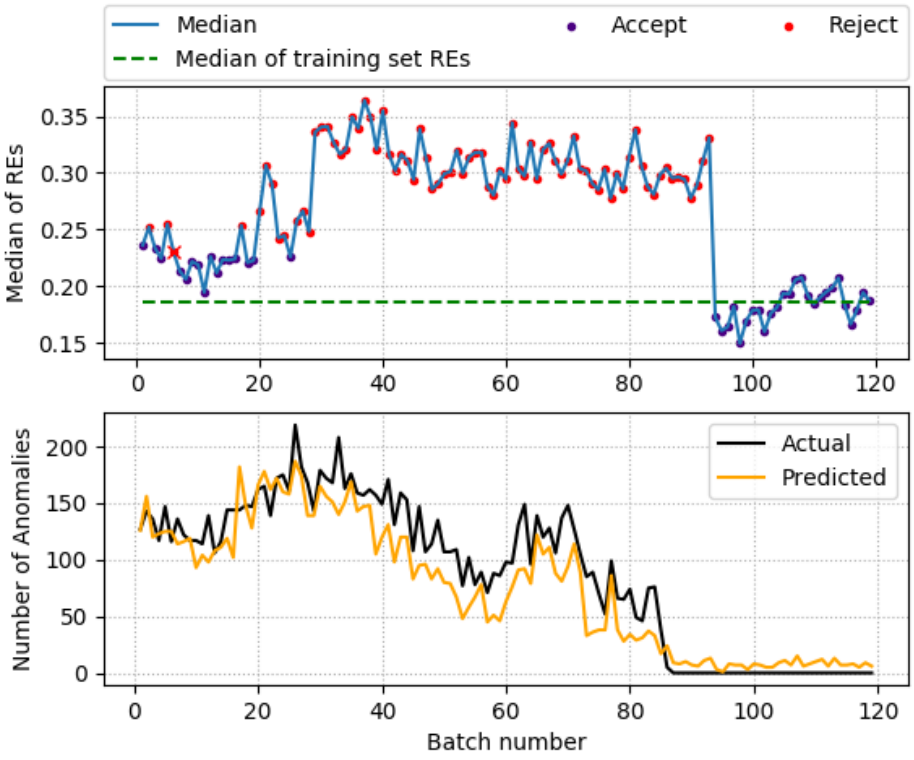


Figure 3.7: Moving median w.r.t Number of Anomalies

a fixed threshold. It allows capturing of new variants of normal data that are within limits of the REs of the expected normal data. Allowing the median value to move freely shifts the median too far from the expected normal data which results in classifying some anomalous points as normal data. Having a fixed threshold, though conservative, does not accommodate variants of new normal data and hence, performs poorer. This is further affirmed by Figure 3.7. Figure 3.7 shows that the median values of the REs of each batch increases when there are anomalies present in each batch. This allows us to understand how different each batch is from the training set. In the last 20 batches, though there are no anomalies, the median value is allowed to drift above the fixed threshold. This gives evidence that new variants of normal data can have higher REs and accommodating this allows for better performance as depicted in Table

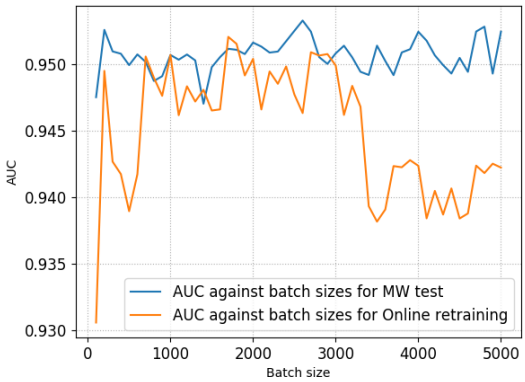
3.5.

The 6^{th} batch indicated with a red 'x' on Figure 3.7 is a median rejected by the MW test even though it is small. This implies that the points in the batch have relatively higher REs despite having a small median value. The value of θ is set at 85^{th} percentile and with batch size of 1000. Even though the number of actual anomalies is less than $(100-85)\% \times 1000 = 150$ from batch 44 onwards, H_0 is rejected for a number of iterations. This implies that there are new variants of normal data in the batch skewing the distribution of RE to a larger median value. This can also be seen in batches 90-93, amongst the last few rejected batches where there are no actual anomalies. These batches in fact contain important variants of normal data that the offline model has not seen before. Further processing could be done on these batches and be used to update the offline model. Any data from batch 95-120 will not be useful in updating the offline model, which has already acquired profiles for these batches.

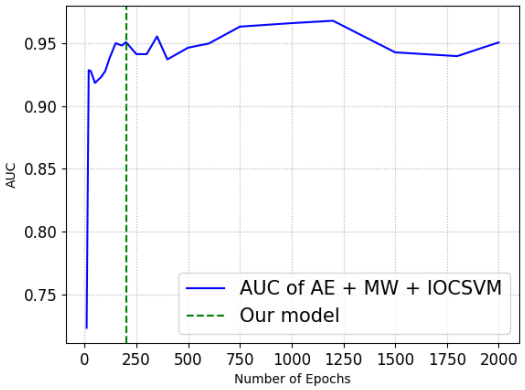
3.2.4 Varying Framework Parameters

Figure 3.8a shows graphs of varying batch size for the MW test and for online training. The MW test is not strongly affected by batch size. For online training however, there is a drop in performance with larger batch sizes. This is because with larger batch sizes, the online model retraining occurs less frequently and the data profiles have changed.

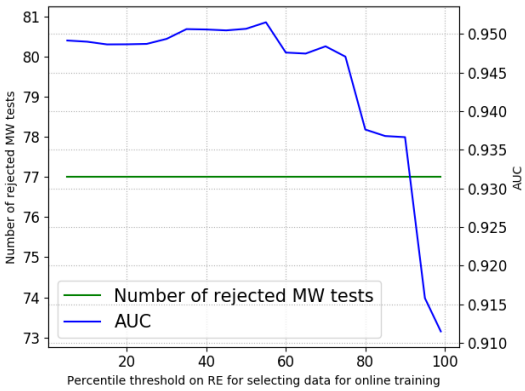
In Figure 3.8b, as the number of epochs increases, there is no real drop in performance of the hybrid AE+MW+IOCSVM model. This is because the role of the offline model in the hybrid is not to give an anomaly score but rather to select points for the online model to learn from. On the UNSW-NB15, the vanilla DAE performs better with average AUC of 0.9098 against the Hybrid DAE+MW+IOCSVM with an AUC of 0.8656. The difference is highest on the CTU13-9 dataset. Training the model with noisy input performs regularisation and it generalises to new data better [19,



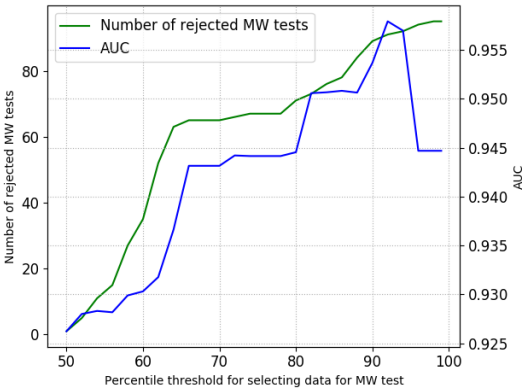
(a) AUC against different batch sizes



(b) AUC against number of epochs



(c) Number of rejected tests against percentile threshold



(d) Number of rejected tests against θ

Figure 3.8: Results varying parameters of the AE+MW+IOCSVM on UNSW-NB15

158]. Thus, the vanilla DAE did well. Dropout is a technique by which units are randomly dropped during training to improve regularisation: if this is included in AE training [178], similar results are obtained on the CTU13-9 dataset where the vanilla AE does better than its hybrid. In Table 3.6, the AE is trained with a dropout probability of 0.5 on its first hidden layer on the CTU13-9 dataset. Even though the offline DAE performed better on average than its hybrid, it was not consistent. In such cases,

this can mean that the model has not generalised correctly or enough, or the new streaming data is different. Also, since it cannot be determined how the model generalises in high dimensional space, one cannot be confident that the model will do well over future data streams. The under-regularised AE avoids this situation all together, learns the training set well and gives a higher RE to any data point that is not similar. The job of anomaly scoring is completely delegated to the online model. The benefit of this approach is that there is less need to focus on training a model to generalise to new data.

Table 3.6: AE trained with dropout on CTU13-9

Method	**AUC**		**Method**	**AUC**
AE+MW+IOCSVM	0.7426		DAE+MW+IOCSVM	0.7954
Offline vanilla AE	0.9385		Offline DAE	0.9677

Figure 3.8c varies the percentile threshold on the RE values for selecting data for online training. The median was used under the assumption that at least 50% of the data is normal. With a higher value, more anomalous points enter online training as normal data, thus the AUC score drops. Any value less than 70^{th}-percentile would be acceptable for the UNSW-NB15 stream. This value depends on the percentage of the expected normal data in each batch, but has no impact on the number of rejected MW tests. This also suggests that in the event of a batch with more anomalies, the model is robust to not allow the threshold $thres$ to increase.

In Figure 3.8d, θ is varied in the range of [50,99] percentile for selecting data for the MW test in the UNSW-NB15. The total number of possible rejected tests is 119. As expected, using a higher θ for the MW test results in more of the batches rejecting the null hypothesis, H_0. This is because anomalous data points which have higher REs are being considered in the batch for the MW test. Having a high θ would yield similar results as a fixed median threshold. Here the median is initialised as described in Algorithm 1 (*cf.* the analysis in Table 3.5). Similarly, having θ too small

allows H_0 to be accepted: the model will then give results similar to a moving median without the MW test as in Table 3.5, resulting in lower AUC as seen in Figure 3.8d. A small threshold of $\theta = 50$ percentile can be used (in parallel) to check whether most of the data in the batch is anomalous, without updating thresholds. Rejection of H_0 on a lower θ strongly suggests that most points in the batch are anomalous. This could signify an intrusion since the lower half of the batch's REs leads to rejection of H_0.

The main concept behind the improvement of the AUC is due to being able to correctly identify inliers which are normal data in the online model that the offline model has identified as anomalies as depicted in Figure 3.9.

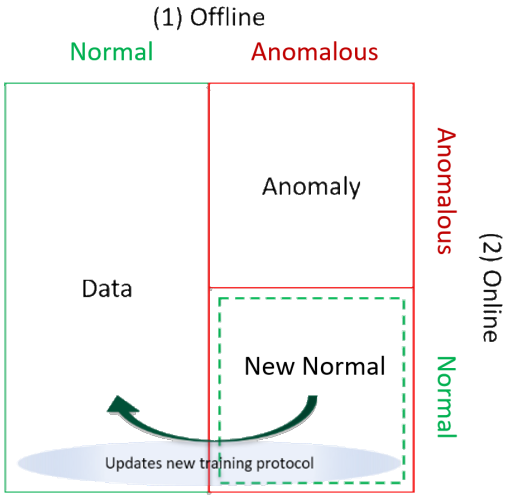


Figure 3.9: Online model identifying new normal data.

3.3 Detecting New Data

One key difference from other methods in the literature is that this offline model learns the training set well and trades bias for higher variance, which prevents generalising incorrectly on new data. Using the MW-test, it is determined whether the streaming data is suspiciously different from the training set. This is possible because it is the online model which per-

forms the anomaly scoring. The training and threshold update can be done in parallel, so scoring is not delayed in live streams. Following that, the heuristic to detect new data is found in Algorithm 2.

Today's anomalies may be tomorrow's normal. With new types of innovations constantly occurring, new legitimate data especially in network traffic are bound to occur. Anomaly detection systems that perform well on current data will most likely flag these new patterns as anomalies. The main idea for detecting them is to look for clusters amongst anomalous data that remain after removing current normal data. This works better when the new type of data fall close to each other in the same spectrum of anomaly scores. In a well regularised model, this tends not to be the case. In the hybrid framework, as anomaly score of 0 is applied to all data less than $thres$, these data can be immediately removed and thus, reduce the search space (lines 2-3, Algorithm 2). Another important primary step is to remove any preprocessing (Appendix B.1.1) done on the data based on existing normal data (line 5, Algorithm 2). This step could be done continuously but it is most effective once enough data are obtained.

The range of anomaly scores is first split into t intervals (lines 12-14, Algorithm 2). Then clusters are searched for by removing points as the range is limited (lines 15-16, Algorithm 2). Principal component analysis (PCA) is performed to reduce the dimensionality here. Note that the offline model cannot be used to do this because it is trained on existing normal data. Clustering algorithms such as DBSCAN will do well here (*cf.* their use in [36, 46]) as the aim is not to spot outliers but clusters (lines 21-22, Algorithm 2). There is no restriction on the number of intervals or their sizes.

This algorithm embodies an assumption that new normal data consists of connected dense regions amongst the points with high anomaly scores, building on the intuition that new normal data does not attempt to hide or mask in the way that anomalies might. The value of M is not deterministic however, making this is a heuristic rather than a definitive method. It is

Algorithm 2 Search for new normal data. All acronyms are found in Table 3.1.

```
while True do                          ▷ Stream data x_i with anomaly scores
    counter ← 0
    if Anomaly sc_i == 0 then
        Drop x_i
    else
        x_i ← RemovePreprocessing(x_i)
        counter += 1
    end if
end while
```

```
if counter == M then
    minval ← min(Anomaly scores)
    maxval ← max(Anomaly scores)
    diff ← (maxval − minval)/t
    for r := 1 to t do
        thres_new ← minval + r × diff
        for i := 1 to M do
            if Anomaly sc_i > thres_new then
                Temp-DB.insert(x_i)
            end if
        end for
        Temp-DB ← PCA(Temp-DB)
        clusters ←Cluster(Temp-DB)
        NewNormalData ←SpotClusters(clusters)
        Temp-DB←Empty(Temp-DB)
    end for
end if
```

a challenge to automatically determine the size of a cluster required for it to be deemed "normal" [185]. On a positive note, the search space is smaller and the problem is inverted: instead of looking for non-clustering points in a sea of data, looking for clustering points in subsets of the sea of anomalies is a more manageable task.

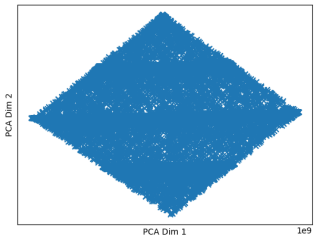
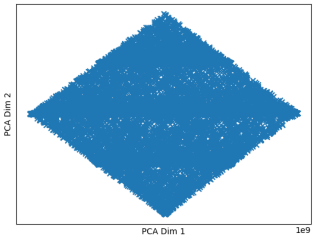
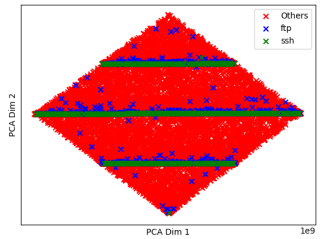
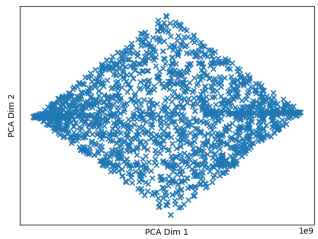
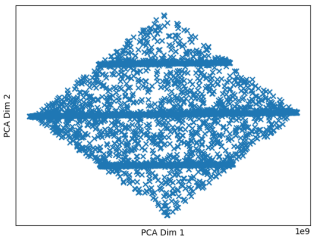
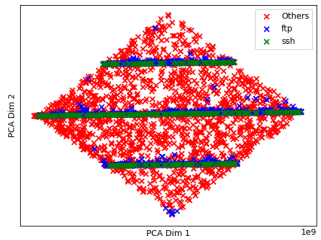
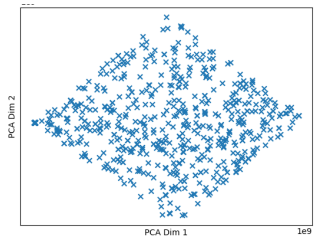
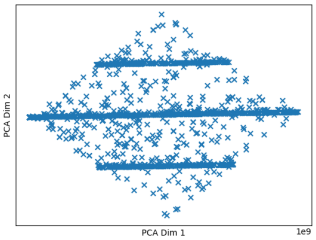
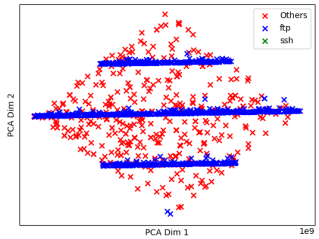
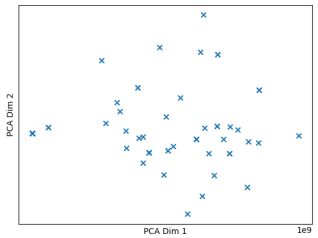
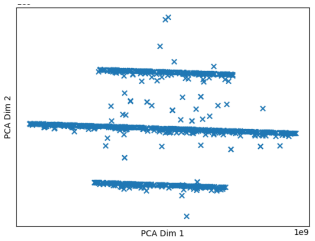
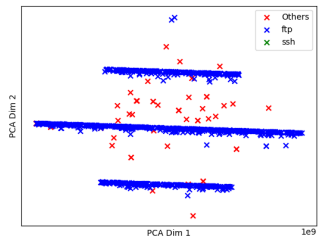
3.3.1 Performance

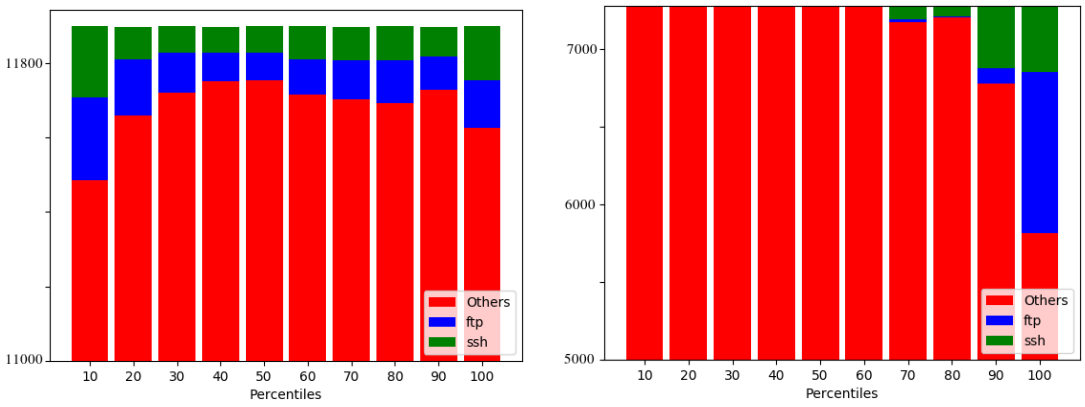
To detect new normal data, the anomaly detection system must discriminate all of them as anomalous and rank them close to each other. In a well regularised model this is not possible, because the new data type may be given varying anomaly scores. The anomaly scores of the offline vanilla VAE (which obtained the best AUC score of 0.9345 on the UNSW-NB15-New stream), are compared against the AE+MW+IOCSVM which is the best overall performing model. Here 'ftp' and 'ssh' data are considered new because they were excluded from the training data. In Figure 3.10 the anomaly scores are separated into 10 percentiles and the data type in each percentile range is shown. There are fewer points in the hybrid because scores equal to 0 have been dropped (line 3, Algorithm 2). This is not possible with the offline VAE to reduce the search space. This benefit is exclusive to the hybrid framework.

Looking at the VAE, the new data falls over all percentile ranges while for the hybrid model, the new data falls in the higher end of the spectrum. The hybrid model scores the new data much higher than the model which generalises better to new data. Hence, it will be easier to identify clusters on the scores of the hybrid model. This phenomenon occurs for all of the models in the Hybrid framework.

To bring out the clusters as specified in line 21 of Algorithm 2, results of PCA in 2 dimensions of the hybrid AE+MW+IOCSVM (line 20, Algorithm 2) is shown. With $t = 10$ in Algorithm 2, Table 3.7 shows the PCA plots for anomaly scores greater than $thres_{new}$ for r = 1, 5, 6 and 10. When

Table 3.7: PCA plots for anomaly scores greater than each $thres_{new}$. $thres_{new}$ is increased for each subsequent plot downwards for $r = 1$, 5, 6 and 10 respectively. Green, blue and red points indicate 'ssh', 'ftp' and other types respectively in the ground truth column.

Without new data	**With new data**	**Ground truth**
UNSW-NB15	UNSW-NB15-New	UNSW-NB15-New
		
		
		
		



(a) offline VAE (b) AE+MW+IOCSVM

Figure 3.10: Data types in each percentile range on UNSW-NB15-New.

there is new data, as $thres_{new}$ increases, when going down the columns, the connected dense regions become clearer. This phenomenon does not occur when there are only anomalies.

3.4 Binary Classification

The hybrid framework can also be used for binary classification instead of one-class classification [141, 143]. As both normal and anomaly classes are required during training, this is considered a Signature-based model. An example model using Radius Nearest Neighbour (For brevity this shall be referred to as 'Rad-NN' in what follows) and SVM is presented in this section.

3.4.1 Model Description

Figure 3.11 depicts the system overview. x_i denotes the data to be classified. The offline model is trained offline with labelled data. The online model is trained with recent data that has traversed through the network and their corresponding labels are as accurate as the confidence of the sys-

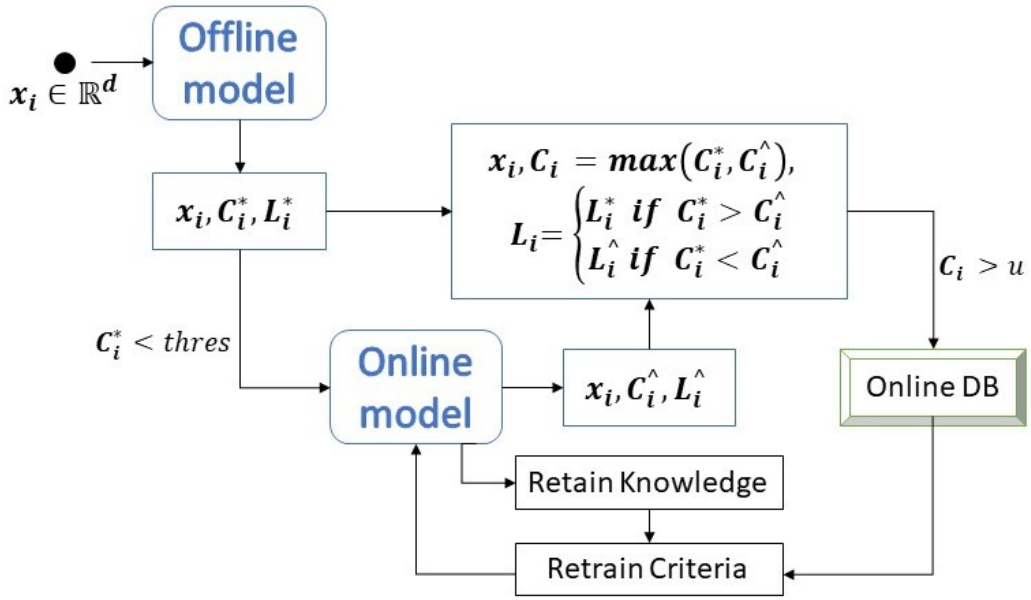


Figure 3.11: System overview, C denotes the confidence measure of the prediction label L.

tem. x_i is classified by the offline model with confidence C_i^*. If C_i^* is lower than a specified threshold, $thres$, or the offline model is unable to classify it $(C_i^* = 0)$, the online model is invoked to classify x_i with confidence $C_i\hat{}$. The label, L_i then depends on the model with the higher confidence, $C_i = max(C_i^*, C_i\hat{})$. Thereafter, the data classified by the system with high confidence, $C_i > u$ for an upper threshold u is obtained to retrain the online model. The online model will retrain itself by retaining some knowledge and using high confidence points when a retraining criteria is met. To implement this system, Rad-NN as the offline model and the SVM as the online model are chosen.

Nearest Neighbour models are also known as 'lazy learners' because they do not learn a discriminative function but "memorize" the dataset [154]. They simply maintain a knowledge base of points in hyperspace and when tasked with classifying a new point, x_i, the model calculates the distance between x_i and its neighbours. It then counts the votes or classes

of the neighbouring points and classifies x_i. This is an appropriate model to retain knowledge or useful points that encompass the general features of network data. The voting can be done based on equal weighting or distance weighting where closer points have a higher weight on their vote for x_i. Equal weighting might not be appropriate in the presence of class imbalance. The commonly used k-NN model does this based on a user defined value k for the number of nearest neighbours to vote. An issue with k-NN is that sometimes a point might be quite far from all of the other points and yet it is able to classify the new point. Rad-NN counts the votes of neighbouring points within a specified radius. In this context, if x_i is too far away, it will be an outlier and the model will be unable to classify it. The complexity for the search is of $O(d \times log_2(n))$, where n is the number of points and d is the number of features with an implementation of a *Ball tree* structure.

Upon classification, the confidence C_i^* of the Rad-NN's classification of point x_i is calculated using the following function.

$$C^*(x_i, k) = 1 - \frac{\sum_{j=1}^{k} D(F, NLN_j(F))}{\sum_{j=1}^{k} D(F, NUN_j(F))} \tag{3.1}$$

$D(a, b)$ represents the distance between points a and b. $NLN_j(F)$ represents the j^{th} nearest like neighbor of x_i while $NUN_j(F)$ represents its j^{th} nearest unlike neighbor [78]. The number of like points or unlike points are not constrained by the value of the radius. Since Rad-NN uses distance weighting for classifying, this measure will always be less than 1. This confidence scoring addresses two points. Firstly, the closer the target data point is to unlike points, the denominator value will be low and the confidence score will be low. Secondly, if the target data point has few like neighbours close to it, then it is occurring in a sparse region. The numerator value will be high as it would have to take the distances of like

points farther away and so the confidence score will be low. This scoring methodology is independent of the classification methodology.

For the online model, SVMs are used as they are simple and efficient machine learning models in high dimensional spaces. They use the kernel trick to map data into a higher dimensional subspace where a decision boundary, given by a decision function, is obtained [20]. SVMs can be trained with limited data. As normal data changes, SVM can incrementally learn with the new data and shift its decision boundary. The Radial Basis Function (RBF) kernel is chosen because it does not assume any prior knowledge about the data, is invariant to translation and is able to fit every target value exactly [20]. Furthermore, the bias and variance can be adjusted using the C and γ parameters in the RBF kernel. C is the penalty parameter. A low value of C will give us a simpler decision function at the cost of training accuracy. γ is the inverse of the influence of each training point on the decision function and adjusts the spread of the decision region. For the hybrid model, the SVM can have high variance because it is trained on high confidence points. Training time is $O(max(n,d) \times min(n,d)^2)$ [32]. For a small number of samples, this does not pose an issue. Furthermore, once the SVM is trained, the support vectors are enough to describe the current decision boundary. These support vectors will be retained for the next retraining in the system. The SVM will be retrained in batches when the number of high confidence points in the database reaches a specified number. In this way, as depicted in Figure 3.12, the decision boundary of the SVM will shift as it is retrained with new data.

For the SVM, its confidence $C\hat{}$ can be computed using Platt scaling on the scores of the decision function. Platt scaling is an algorithm to give a probability estimate on the output of the decision function. This probability is calibrated using the data points used to obtain the decision function [105].

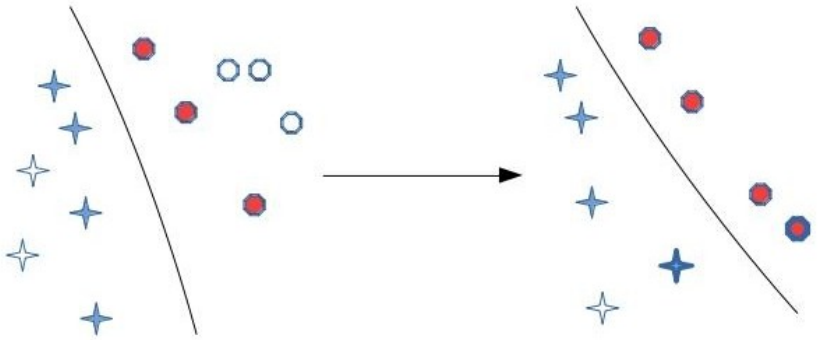


Figure 3.12: SVM decision boundary shift. The shaded items represent support vectors. Support vectors are retained on the right diagram with the addition of two new points represented by thickness. The decision boundary shifts slightly and new support vectors are obtained.

3.4.2 Performance and Evaluation

This model is evaluated on the NSLKDD dataset [43]. Data preprocessing and model training details including grid-search for determining the parameters are found in Appendix B.2.1. The SVM parameters are $C = 100$, $\gamma = 0.1$. Confidence threshold, $thres$ for Rad-NN is 92.2%. k, to calculate its confidence is 10. Upper threshold, u for online training selection is 96%. The SVM is retrained when the number of previous support vectors and high confidence points reaches 1300. The results are as follows. The system achieves 95.55% accuracy, 94.42% detection rate and 2.96% false positive rate. The SVM was trained 11 times whilst predicting 21,244 data points in the testing set.

Table 3.8 compares the hybrid system with other offline + online algorithms implemented in a similar fashion. The Gaussian Naive Bayes (GNB) is used as the other online algorithm. With the GNB, all online models are retrained completely with no points being retained as there is no clear way to retain points for the GNB. The classification probabilities of GNB is taken as its confidence measure. In Table 3.9, the hybrid system with other offline methods. In Table 3.10, the hybrid system is compared

with other online methods.

To test the system further, the offline model is trained without each one of the anomaly types from the dataset. The respective anomaly is also removed from the points used to initialise the online model. The results are found in Table 3.11. A similar experiment in Table 3.12 is shown but the online model is initialised with all anomaly points.

Table 3.8: Comparison with other offline and online methods

Model Offline + Online	**FPR (%)**	**TPR (%)**	**Acc (%)**
Rad-NN + SVM	**2.96**	**94.42**	**95.55**
Rad-NN + SVM[a]	3.80	88.82	92.00
Rad-NN + GNB	3.95	81.01	87.47
SVM + GNB	8.15	70.68	79.78

[a]Support vectors are not retained.

Firstly, retaining the support vectors gives better overall results compared to Rad-NN + SVM without retention in Table 3.8. Training without retention runs the risk of shifting the decision boundary too much, especially more since the data points are only as accurate as the confidence label.

Offline methods are not effective in capturing concept drift and higher detection rates comes at the expense of higher false positive rates as seen in Table 3.9. The Rad-NN by itself gives us the poorest detection rate. This is certainly expected as its role is to capture the general features of the data. The SVM trained offline also does not perform well because it was trained with too many irrelevant points, i.e. noise. Though they do not perform as well individually but together, they are able to overcome their inadequacies. The performance of Rad-NN is improved when complemented with an online model in Table 3.8. However, the SVM trained

Table 3.9: Comparison with offline methods

Model	**FPR (%)**	**TPR (%)**	**Acc (%)**
Rad-NN + SVM	2.96	94.42	**95.55**
SVM (C=100, γ=0.1)	8.04	71.01	80.02
Rad-NN	**2.13**	59.33	75.90
Decision Tree	3.37	70.12	81.51
Gaussian Naive Bayes	3.26	62.04	76.95
k-NN (k=5,equal-weighted votes)	2.88	61.33	76.71
Artificial Neural Networks [83]	3.23	81.20	81.16
Deep Recurrent Neural Networks [213]	3.07	72.95	83.28
Self taught learning [85]	21.60[a]	**95.95**	88.39
Stochastically improved Denoising Autoencoder [11]	4.01[a]	83.08	88.65

[a]Derived from the confusion matrix of the published results

Table 3.10: Comparison to other online methods

Model	**FPR (%)**	**TPR (%)**	**Acc (%)**
Hybrid System	**2.96**	**94.42**	**95.55**
SOM & Neural Network [79]	14.06	94.40	90.00
2 layer GMM clustering [18]	7.00	85.00	88.44[a]
Deep Belief Network with Adaptive Linear Function [8]	4.47[a]	95.2	95.34[a]

[a]Derived from the confusion matrix of the published results

Table 3.11: Detection after removing each type of attack during offline training and online initialisation

Training Data	% R2L detected	% DoS detected	% Probe detected	% U2R detected
Full System	78.39	99.60	97.68	85.03
Data without R2L	14.67	99.31	97.64	81.28
Data without DoS	73.89	79.69	98.52	83.42
Data without Probe	77.55	99.57	61.50	86.63
Data without U2R	77.93	99.60	97.95	80.21

Table 3.12: Detection after removing each type of attack during offline training only

Training Data	R2L % detected	DoS % detected	Probe % detected	U2R % detected
Full System	78.39	99.60	97.68	85.03
Data without R2L	73.66	99.60	97.42	81.81
Data without DoS	73.36	99.21	98.47	85.56
Data without Probe	77.25	99.44	93.28	86.10
Data without U2R	78.01	99.60	97.82	84.49

purely offline shows better performance than SVM + GNB. This is mainly because the offline SVM misclassifies with higher confidence than the Rad-NN and more incorrectly labelled points are used in the training of the online model. Also, since the confidence scoring of Rad-NN is independent of its classification methodology, the Rad-NN does not face this issue. This suggests that a strong offline classifier with a robust confidence scoring methodology is necessary.

Comparing Table 3.10 with Table 3.9, online methods have higher overall detection rates but they also have higher false positive rates. This is because it is difficult to optimise in real time. Online models are better at capturing concept drift and so they have higher detection rates.

In Table 3.11, after complete removal of each type of attack, the model still detected more than 50% of DoS, Probe and U2R data points as anomalies. However, it was only able to detect 14.67% of the R2L attacks. This is because R2L attacks are similar to normal data and possibly closer to the boundary. Figure 3.13 shows a PCA analysis of the testing set. In two dimensions, R2L and normal data points are more agglomerated compared to the other anomalies. In Table 3.12, the online model had a head start with a few points of each attack trying to mimic the situation where the online model has already detected a new attack before. In such a scenario, there is a huge improvement in detecting each type of attack compared to Table 3.11. Hence, inserting a new type of anomaly into the online model during one of its retraining is sufficient for the system to adapt and learn to detect the anomaly in future.

In Tables 3.11 and 3.12, the system trained without certain attacks enables better detection of other anomalies while poorer detection for some others. For example in Table 3.11, without DoS attacks, Probe attacks have a higher detection rate while U2R attacks have a lower detection rate. There may be a few reasons for this phenomenon. Firstly, with fewer unrelated anomalies in the RadNN, there is less noise. On the contrary, the misclassification comes at a cost of training the SVM incorrectly. Fur-

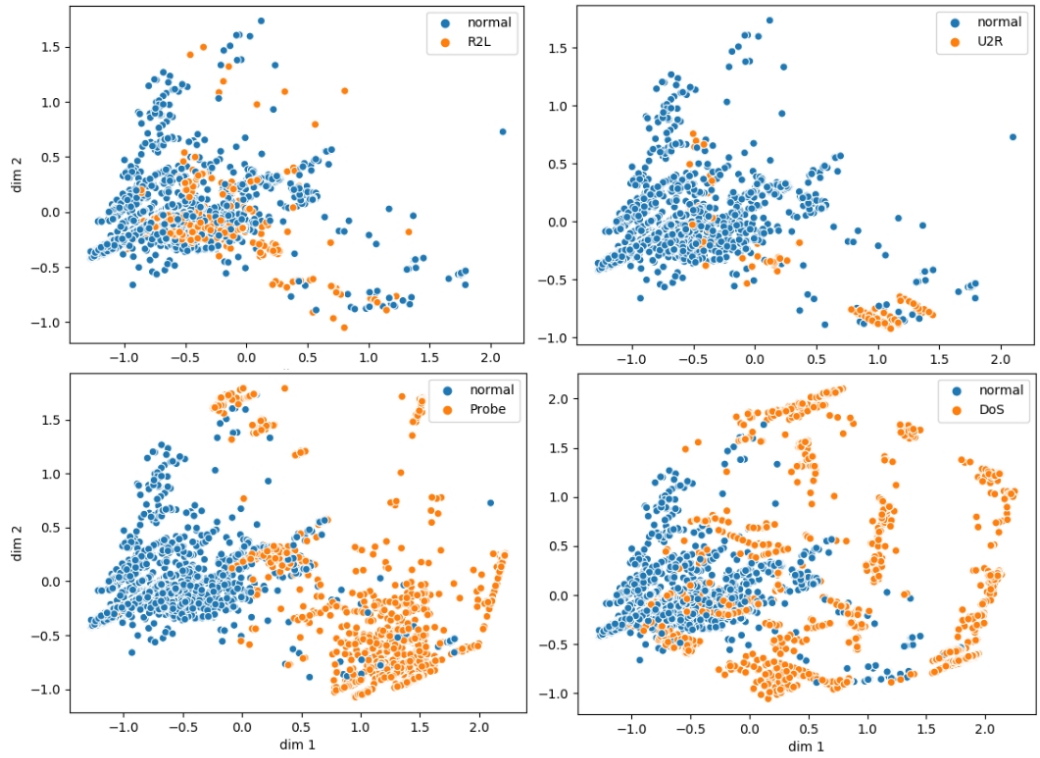


Figure 3.13: PCA analysis of testing set

thermore, the SVM is overfit with a high penalty parameter C in the RBF kernel. The fluctuations are not consistent because part of the detection process depends on the shifting decision boundary of the online SVM and this depends on the recent points that have traversed through the system.

A few caveats of this model are as follows. This model is for binary classification and labelled anomaly and normal points are necessary for training. Furthermore, this model does not consider the curse of dimensionality unlike the AE in the behaviour-based hybrid framework.

Performance on Streaming Data

Streaming data is not a requirement for this model because the assumption that most of the data is normal is not used. Regardless, to make it

complete, the performance of the model is evaluated on the UNSW-NB15 streaming dataset as described in Section 3.2 which is of streaming nature and the results are shown in Table 3.13. The training dataset is used to train the model.

Table 3.13: Rad-NN + SVM on UNSW-NB15 stream

Dataset	**FPR (%)**	**TPR (%)**	**Precision (%)**	**Acc (%)**
UNSW-NB15 Stream	2.16	98.74	81.78	97.92

3.5 Conclusion

Both the behaviour based hybrid framework and the binary hybrid model have been shown to perform better than an offline or online model individually. It is robust under evolving data. The framework also allows one to identify new normal data as dense connected regions amongst anomalous data. The binary model can be used when the type of anomalies are known. Chapter 6 will highlight avenues for improvement.

Continuing to address the challenges highlighted in Chapter 2, the next chapter looks at how anomaly detection can take into consideration distributed data and ties it in with the hybrid framework developed here.

Chapter 4

Distributed Training

With the advent of *Big Data*, pooling the data at one location incurs a high communication overhead. In wireless transmissions, the problem is further exacerbated. This chapter addresses the training of a model when the data is heterogeneously distributed. Moreover, with Edge AI, in some real-time applications such as autonomous vehicles, both training and inference need to be made quickly as well.

In this aspect, Bayesian models have many advantages as mentioned in Section 2.2 but the methods in the literature have not considered a fully Bayesian approach. This chapter will show the use of Expectation Propagation (EP) to build a Bayesian Random Vector Functional Link AutoEncoder (BRVFL-AE) in a distributed manner. The BRVFL-AE has a closed form solution. It is also a SLFN and thus, it has lower computational and memory complexity. It is then compared to a standard Bayesian AutoEncoder (BAE) which requires Variational Inference (VI) to build the posterior (Section 2.2.1).

4.1 Preliminaries

This section introduces the Bayesian Neural Network (BNN) used at edge sites, and the EP algorithm used to combine information [164, 190]. More

details on Bayesian methods for regression are found in Appendix C and Bishop [20].

4.1.1 Bayesian Random Vector Functional Links

Let $\mathbf{x} \in \mathbb{R}^d$ be the input vector. The output of a neural network with one hidden layer is denoted by

$$f(\mathbf{x}) = \sum_{l=1}^{B} w_l h_l(\mathbf{x}) = \mathbf{w}^T \mathbf{h}(\mathbf{x}) \tag{4.1}$$

where $h_l(\cdot)$ as shown in 4.2 is a randomly initialised non-linear mapping which project the input to a higher B dimensional space. h_l is the classical sigmoid function with weights $\mathbf{a} \in \mathbb{R}^B$ and biases $b \in \mathbb{R}$ drawn from a uniform distribution in the range $[-\lambda, \lambda]$ where λ is a positive real number as shown in [164].

$$h_l(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a}^T \mathbf{x} + b))} \tag{4.2}$$

Structuring the above problem as a ridge regression problem with dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1 \dots n\}$ for $y_i \in \mathbb{R}$, the optimal weights $\mathbf{w}$ are found by solving

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^B}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\mathbf{H}\mathbf{w} - \mathbf{y}\|^2 + \frac{C}{2} \|\mathbf{w}\|^2 \right\}. \tag{4.3}$$

In Equation (4.3), $\mathbf{H}$ is built by stacking row-wise vectors $\mathbf{h}(\mathbf{x}_i)$ and C is the regularisation factor. The solution can be found analytically via the Moore-Penrose inverse:

$$\mathbf{w}^* = (\mathbf{H}^T \mathbf{H} + C\mathbf{I})^{-1} \mathbf{H}^T \mathbf{y}. \tag{4.4}$$

Considering a multidimensional output instead, replacing y_i by $\mathbf{y}_i \in \mathbb{R}^d$, where each dimension is independent, a multidimensional regression problem is obtained where the equations are as before but $\mathbf{w}$ is now $B \times d$ and is independent along each dimension $j = 1, \dots, d$. This independence formulation is possible as only one set of weights are being trained.

Let X denote the data. $P(X|\mathbf{w})$ denotes the likelihood and $P(\mathbf{w})$ denotes the prior. The posterior distribution over the parameters is given by Bayes law:

$$P(\mathbf{w}|X) \propto P(X|\mathbf{w})P(\mathbf{w}). \tag{4.5}$$

From the results for Bayesian ridge regression [20, 164], let each of the outputs vary by white noise with variance σ^2, resulting in (4.6) for the likelihood in each dimension. $\mathcal{N}(x|a, b)$ denotes a Gaussian distribution over x with mean a and variance b. For the prior, allow the weights to be small and express it as a zero-mean multivariate Gaussian with diagonal covariance. In Equation (4.7), $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{S}_0 = \gamma^{-1}\mathbf{I}$ for all j and γ is the precision parameter.

$$P(X|\mathbf{w}_j, \sigma^2) = \mathcal{N}(x_{ij}|\mathbf{w}_j^T\mathbf{h}(\mathbf{x}_i), \sigma^2) \tag{4.6}$$

$$P(\mathbf{w}_j|\mathbf{m}_0, \mathbf{S}_0) = \mathcal{N}(\mathbf{w}_j|\mathbf{m}_0, \mathbf{S}_0) \tag{4.7}$$

The posterior is again Gaussian, with mean $\mathbf{m}_j$ and covariance $\mathbf{\Sigma}$ given by

$$\mathbf{m}_j = \mathbf{\Sigma}(\mathbf{S}_0^{-1}\mathbf{m}_0 + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{x}_j) \tag{4.8}$$

$$\mathbf{\Sigma} = (\mathbf{S}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H})^{-1} \tag{4.9}$$

If σ^2 is constant for each j then $\mathbf{\Sigma}$ depends only on the variation in the data given by $\mathbf{H}^T\mathbf{H}$. The predictive distribution is also Gaussian,

$$P(\hat{\mathbf{x}}_j|\hat{\mathbf{x}}, \sigma^2, \gamma) = \mathcal{N}(\hat{\mathbf{x}}_j|\mathbf{m}_j^T\mathbf{h}(\hat{\mathbf{x}}), \phi(\hat{\mathbf{x}})^2) \tag{4.10}$$

$$\phi(\hat{\mathbf{x}})^2 = \sigma^2 + \mathbf{h}(\hat{\mathbf{x}})^T\mathbf{\Sigma}\mathbf{h}(\hat{\mathbf{x}}). \tag{4.11}$$

The method above provides not only an estimate for the mean of the weights, but also a variance for the weights and therefore, a predictive variance for $\mathbf{y}$. Furthermore, the solution is in closed form. Instead of specifying values for the hyper-parameters σ^2 and γ, a conjugate prior distribution can be placed on them to perform Bayesian inference and obtain MAP values from the data.

4.1.2 Expectation Propagation

EP was formally introduced by Opper and Winthler [146] and generalised by Minka [120]. It is an iterative message-passing algorithm which minimises the Kullback-Leibler (KL) divergence of the reverse form: KL$[P(\mathbf{w}|X)||q(\mathbf{w}|\theta)]$ between two distributions [20, 100]. From (4.5), if there is no closed form solution, VI [20] or MCMC can be used to estimate the posterior or to sample from the posterior. It is then projected on to q, a member of the family of exponential distributions, by matching its moments [20, 166]. There is no guarantee of convergence but it has been shown to work well for models with log-concave factors such as the Gaussian distribution [65]. In the case where the posterior is in the exponential family, minimizing the divergence conveniently corresponds to matching the moments [120].

As in FL, if a global model exists, the IID assumption is invoked with respect to the global model for the data, though they may be unevenly balanced between sites. This may also be viewed as performing Bayesian inference site by site taking the posterior as the new prior in the subsequent iteration. Else, other alternatives such as learning distinct local models should be considered [104, 174]. From (4.5), the likelihood is factored into partitions, one for each site in the network. These are then combined iteratively with an approximate prior to produce the global posterior upon convergence. As shown in 4.12, let there be $k = 1, \ldots, K$ sites and X_k denote the data in the k^{th} edge site. The dimension variable j is omitted to reduce clutter.

$$P(\mathbf{w}|X) \propto \prod_{k=1}^{K} P(X_k|\mathbf{w})P(\mathbf{w}) \tag{4.12}$$

is approximated by

$$g(\mathbf{w}|\mathbf{r}, \mathbf{Q}) \propto \prod_{k=1}^{K} g_k(\mathbf{w}|\mathbf{r}_k, \mathbf{Q}_k) g_0(\mathbf{w}|\mathbf{r}_0, \mathbf{Q}_0), \tag{4.13}$$

where $g(\mathbf{w}|\mathbf{r}, \mathbf{Q})$ is a member of the exponential family and $\mathbf{r}, \mathbf{Q}$ are the

natural parameters. Details on exponential family of distributions are found in Appendix C.1.1 and [38]. The exponential family of distributions is closed under multiplication and translates to addition of the natural parameters [38]. The EP algorithm is stated as follows. Firstly, initialise $\mathbf{r}_k, \mathbf{Q}_k = 0$ and $\mathbf{r}_0, \mathbf{Q}_0$ as the natural parameters of the distribution at each site and of the global prior respectively. Then $\mathbf{r} = \mathbf{r}_0 + \sum_{k=1}^{K} \mathbf{r}_k$ and $\mathbf{Q} = \mathbf{Q}_0 + \sum_{k=1}^{K} \mathbf{Q}_k$. In the following steps, g_{-k} and $g_{\backslash k}$ are representations for the cavity and tilted distribution as calculated with respect to Equation (4.13).

E1: At each site, determine the cavity distribution g_{-k} by substituting $\mathbf{r}_{-k} = \mathbf{r} - \mathbf{r}_k$, $\mathbf{Q}_{-k} = \mathbf{Q} - \mathbf{Q}_k$.

E2: At each site, approximate the tilted distribution $g_{\backslash k}$ where $g_{\backslash k}(\mathbf{w}) \propto P(X_k|\mathbf{w})g_{-k}(\mathbf{w})$, using VI [20] if no closed form solution is available, and by matching moments.

E3: At each site, compute the change in distribution.
$\Delta\mathbf{r}_k = \mathbf{r}_{\backslash k} - \mathbf{r}_{-k} - \mathbf{r}_k$ and $\Delta\mathbf{Q}_k = \mathbf{Q}_{\backslash k} - \mathbf{Q}_{-k} - \mathbf{Q}_k$

E4: At each site, update the distribution with a damping factor $\delta \in (0, 1]$, $\mathbf{r}_k \leftarrow \mathbf{r}_k + \delta\Delta\mathbf{r}_k$ and $\mathbf{Q}_k \leftarrow \mathbf{Q}_k + \delta\Delta\mathbf{Q}_k$

E5: In a central site, update the global parameters, $\mathbf{r} \leftarrow \mathbf{r} + \delta \sum_{k=1}^{K} \Delta\mathbf{r}_k$ and $\mathbf{Q} \leftarrow \mathbf{Q} + \delta \sum_{k=1}^{K} \Delta\mathbf{Q}_k$

Repeat steps **E1-E5** until $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are small or when the tilted distribution at each site is consistent with the approximate posterior [65]. Assuming each of the weights of the BRVFL-AE follows a Gaussian distribution as in [164], EP can be used in closed form to compute the tilted distribution. Since Gaussian distribution is a member of the exponential family of distributions, there is a one-to-one mapping between the natural parameters and the mean and covariance of the Gaussian distribution:

$$\mathbf{r} = \mathbf{\Sigma}^{-1}\mathbf{m} \qquad \mathbf{Q} = \mathbf{\Sigma}^{-1} \tag{4.14}$$

4.2 EP-BRVFL-AE

How EP can be used in a BRVFL-AE is described here. The main method is firstly formulated for the case where there exists a central site, and then extended to a completely distributed scenario. The hyper-parameters are estimated likewise. Subsequently, the measures to perform anomaly detection are described, and a system diagram is given for implementation. The rationale for each component is summarised in Figure 4.1.

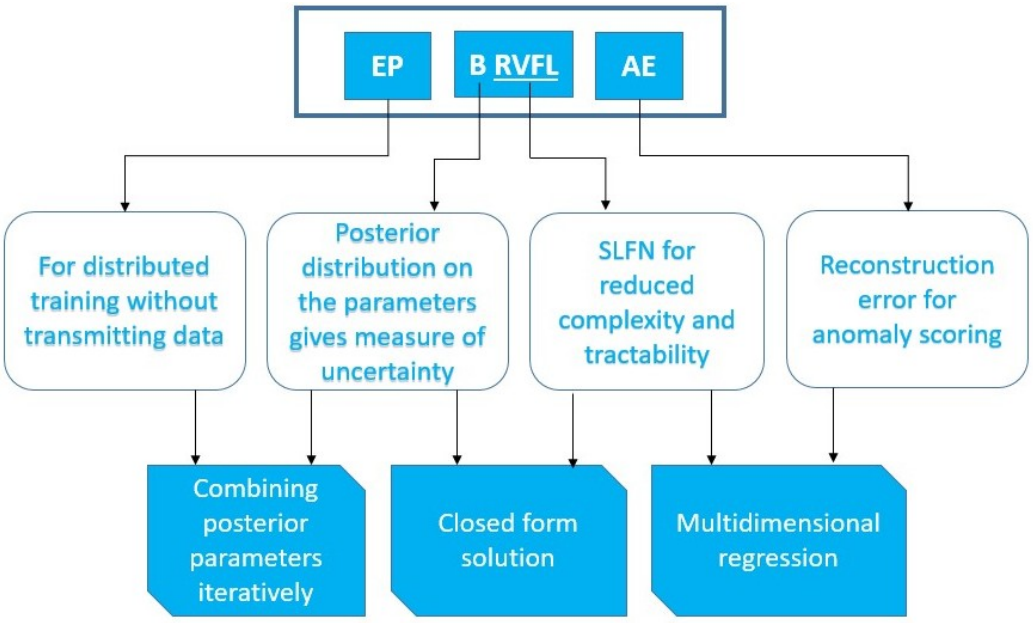


Figure 4.1: Rationale for EP-BRVFL-AE

4.2.1 Central Site

The formulation of EP-BRVFL-AE follows naturally from the methods in Sections 4.1. An AE consists of an encoder network, a latent layer and a decoder network. The encoder maps the input data into the latent layer and the decoder reconstructs them. For the RVFL, the encoder network is fixed and maps the input data into a higher dimension. Skip connections

Algorithm 3 EP-BRVFL-AE(C)

Initialise site and global parameters, $\mathbf{r}_k, \mathbf{Q}_k, \mathbf{r}, \mathbf{Q}$

while $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are large **do**

At each site, k: ▷ Site Operation

  Update site distribution as in **E4** ▷ From 2^{nd} iteration

  Determine cavity distribution as in **E1**

  Compute tilted distribution with (4.8) and (4.9) using cavity as prior

  Compute $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ as in **E3**

  Send $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ to central

At Central: ▷ Central Operation

  for $k = 1, \ldots, K$ **do**

    Initialise $\mathbf{Q}_{-k} = 0$, $\delta = 2\delta_0$.

    while $|\mathbf{Q}_{-k}| \leq 0$ **do**

      $\delta \leftarrow \delta/2$

      for $k = 1, \ldots, K$ **do**

        Calculate $\mathbf{r}_k, \mathbf{Q}_k$ as in **E4**

      end for

      Calculate $\mathbf{r}, \mathbf{Q}$ as in **E5**

      Calculate cavity parameter $\mathbf{Q}_{-k}$ as in **E1**

    end while

  end for

  Send $\delta, \mathbf{r}, \mathbf{Q}$ to sites

end while

are not implemented from input to output, as when autoencoding through a SLFN, the model could merely learn the identity function.

Algorithm 3 presents EP-BRVFL-AE(C) with a central site. Site parameters, $\mathbf{r}_k, \mathbf{Q}_k$ are initialised to zero and global parameters $\mathbf{r}$ and $\mathbf{Q}$ are initialised to $\mathbf{0}$ and $\gamma\mathbf{I}$ respectively. The central site holds in memory a copy of the edge sites' parameters as well, to determine the best δ with respect to all sites. In **E2** of each iteration of EP, the prior is the cavity distribution. It is necessary that $|\mathbf{Q}_{-k}| > 0$. Else, the damping factor δ needs to be reduced. When the data is evenly distributed, the value of δ is rarely reduced. As the BRVFL method performs closed-form updates, the algorithm generally converges in as few as two iterations depending on the heterogeneity of distributed data. The experiments also verify both these aspects.

The computation of update, cavity and tilted distribution (and changes) shall be referred to as "Site operation". The update computation at the site begins on the second iteration (line 3). The computation to aggregate the changes and to determine δ shall be referred to as "Central operation" as mentioned in Algorithm 3.

4.2.2 Fully Distributed

Algorithm 4 gives a fully distributed version, EP-BRVFL-AE(D). Each edge site holds a copy of the global parameters and incorporates new information when it is received. The main difference is that the central operation will be performed at the edge site. The updates from each site is broadcast throughout the network.

Firstly, network topology is described as follows. Network density κ is defined in (4.15). The Average Degree Per Site (ADPS) in (4.16) indicates the average number of neighbours per site. E is the total number of edges in the network and K is the total number of sites. A fully connected network has $\kappa = 1$ while a ring network has $\kappa = 2/(K-1)$. The Maximum

Algorithm 4 EP-BRVFL-AE(D) without Central Site

Initialise site and global parameters, $\mathbf{r}_k, \mathbf{Q}_k, \mathbf{r}, \mathbf{Q}$

while $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are large **do**
At each site, k: ▷ Site Operation
    Determine cavity distribution as in **E1**
    Compute tilted distribution with (4.8) and (4.9) using cavity as prior
    Compute $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ as in **E3**

    Broadcast $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$
    Receive $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ from other sites

    Initialise $\mathbf{Q}_{-k} = 0$, $\delta = 2\delta_0$.
    while $|\mathbf{Q}_{-k}| \leq 0$ **do** ▷ Central Operation
        $\delta \leftarrow \delta/2$
        Calculate $\mathbf{r}_k, \mathbf{Q}_k$ as in **E4**
        Calculate $\mathbf{r}, \mathbf{Q}$ as in **E5**
        Calculate cavity parameter $\mathbf{Q}_{-k}$ as in **E1**
    end while
end while

Number of Hops required for any site to communicate in the network is denoted as Max Hops.

$$\kappa = \frac{2E}{K(K-1)} \tag{4.15}$$

$$\text{ADPS} = \frac{2E}{K} \tag{4.16}$$

Figure 4.2: A network with 5 sites is shown. The left image is distributed with $\kappa = 1$ and the right image has a central site.

If a site receives updates $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ from all other sites, it will achieve the same result as having a central site. In other words, EP-BRVFL-AE(C) and EP-BRVFL-AE(D) with $\kappa = 1$ are equivalent if all other parameters are kept constant. An example diagram is shown in Figure 4.2. For networks with $\kappa < 1$, waiting for Max Hops steps to occur before performing an update is ideal, but the training can also proceed with updates as they arrive. In the latter case, an update from an edge site three hops away will only be incorporated on the third iteration. The clear advantage of not waiting is that there is no need for the site to have any knowledge of the network topology. Table 4.1 shows an example of a network and how two of the sites are updated. Since site A is 3 hops away from site E, the first update from site A will reach site E on the 3rd iteration.

Algorithm 4 appears simpler and with less computation per iteration but it requires more communication to reach convergence. This reduction in computation is mainly due to each site being interested only in the δ value that works for itself. The experiments showed that the first update from each site is the most important. This is because only the changes are transmitted and in the first iteration, the flat initial prior allows the data to provide the most significant changes.

	Received updates incorporated into update	
Iteration	**At site C**	**At site E**
2	A1, B1, D1	D1, F1
3	A2, B2, D2, E1	D2, F2, C1
4	A3, B3, D3, E2, F1	D3, F3, C2, B1, A1
⋮	⋮	⋮

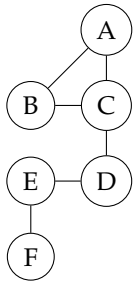


Table 4.1: Numbers represent iterations and letters represent sites, then C1 is the update that is broadcast from site C after the first iteration. C2 consists of updates from A1, B1, D1, and its own data.

4.2.3 Anomaly measures

After Algorithm 3 reaches convergence, the global parameters $\mathbf{r}, \mathbf{Q}$ are used as the parameters for the BRVFL-AE in each site. After Algorithm 4 reaches convergence, each site's global parameters would have converged to a similar value. Anomaly scoring can be performed using one of three measures.

The common scoring method is using the RE for AE. The MAP estimate can be used for the weights. The predictive variance, $\phi(\hat{\mathbf{x}})^2$ can also be used as the measure or the confidence score for RE. Areas with high variance suggest that there is not enough data in the neighbourhood and thus, the point is more anomalous. A heuristic, $\mathcal{H}$, being a combination of both measures is also evaluated.

$$RE = \sum_{j=1}^{d} \|\mathbf{m}_j^T \mathbf{h}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}_j\|^2 \tag{4.17}$$

$$\mathcal{H} = RE \times \phi(\hat{\mathbf{x}}) \tag{4.18}$$

4.2.4 Complexity Analysis

Let c denote the transmission cost of network transmission per parameter, t denote the number of iterations required to obtain δ and s be the number

of EP iterations needed for convergence. For the completely distributed setting, let e_k be the number of neighbours for site k.

Memory

Each site firstly holds n_k data points of dimension d. The parameters $\mathbf{r}$ and $\mathbf{Q}$ contain Bd and $B(B+1)/2$ quantities respectively. In EP-BRVFL-AE(C), at each site, the natural parameters $\mathbf{r}_k, \mathbf{Q}_k$ and the changes $\Delta\mathbf{r}_k, \Delta\mathbf{Q}_k$ are stored. At the central site, the global parameters $\mathbf{r}, \mathbf{Q}$, parameters of all of the sites and the sum of changes of the sites are stored. For EP-BRVFL-AE(D), $4(Bd + B(B+1)/2)$ quantities are stored at each site adding the global parameters and the sum of changes from neighbouring sites. Hence, memory is dominated by $\mathcal{O}(B^2)$.

Computation

For both Algorithms 3 and 4, the cost of computing $\mathbf{H}$ at each site is $n_k(Bd+1)$. At each site, the update, cavity and change computation involving addition or subtraction costs $Bd + B^2$ each. For the tilted distribution, computing $\mathbf{H}^T\mathbf{H}$, $\mathbf{H}^T\mathbf{X}_k$ and $\mathbf{m}$ and inverting $\mathbf{\Sigma}$, cost B^2n_k, Bn_kd, B^2 and B^3 respectively. If $B > n_k$ then the computation cost is dominated by $\mathcal{O}(B^3)$ else, it is dominated by $\mathcal{O}(B^2n_k)$.

For EP-BRVFL-AE(C), summing up the updates costs $K(Bd+B^2)$ at the central site. To compute δ, an additional cost of $2K(Bd + B^2)$ for the update and cavity computation and B^3 to obtain the determinant is required. This computation is repeated t times. For EP-BRVFL-AE(D), summing up the updates is bounded above by $K(Bd+B^2)$, the update and cavity computation costs $2(Bd+B^2)$ and the determinant computation costs B^3. This is repeated t times until δ is determined. In the experiments, if the data is evenly distributed the initial value of $\delta_0 = 1$ suffices.

Communication

The sites transmit the changes in the natural parameters. In EP-BRVFL-AE(C), the central site transmits the global natural parameters with the same cost, $cK(Bd + B(B + 1)/2)$ to all sites. To transmit δ, the cost is cK. In EP-BRVFL-AE(D), the edge sites broadcast the changes to the network. Depending on κ, the communication at each iteration has an upper bound of $cK(Bd + B(B + 1)/2)$.

The total cost is total computation and communication at both edge and central times s. In the experiments, convergence is achieved with $s = 2$ for EP-BRVFL-AE(C) and $s =$Max Hops for EP-BRVFL-AE(D). Considering only the dominating terms, the total computational and communication complexity is $\mathcal{O}(s(cKB^2 + \max_k(n_k B^2, B^3)))$. Amount of data, n_k is under computation and not under communication. With big data, $n_k >> B^2$, this presents significant savings in communication complexity of the network.

Comparison

EP-BRVFL-AE has a lower order of complexity or comparable to other methods. It depends on the number of data items, n_k, linearly, while B is a fixed parameter. For instance, MVE-PCA [147] is cubic in n_k. Table 4.2 gives a summary of these complexities.

4.2.5 Results and Analysis

Performance

The UNSW-NB15, NSLKDD and Australian Credit Approval, Shuttle, Abalone and Pageblocks datasets from the UCI ML repository are used in the experiments. Preprocessing of the data is described in Appendix C.2.

Firstly, changes in the hyper-priors do not yield any significant changes to the AUC score on the UNSW-NB15 dataset. Comparing the different

	Memory	**Communi-cation**	**Computation**
Edge (C)	$\mathcal{O}(B^2 + n_k d)$	$\mathcal{O}(cB^2)$	$\mathcal{O}(B^3 + n_k B^2)$
Central (C)	$\mathcal{O}(KB^2)$	$\mathcal{O}(cKB^2)$	$\mathcal{O}(tB^3)$
Edge (D)	$\mathcal{O}(B^2 + n_k d)$	$\mathcal{O}(cKB^2)$	$\mathcal{O}(tB^3 + n_k B^2)$
MVE-PCA [147]	$\mathcal{O}(d^2 + n_k d)$		$\mathcal{O}(n_k^3)$
Hyperellipsoidal Clustering [112][a] [b]	$\mathcal{O}(\rho_k d^2)$		$\mathcal{O}(\rho_k^2 p + n_k + \rho_k d^3)$

[a] p is the number of nearest ellipsoidals
[b] ρ_k is the number of ellipsoidals at each site

Table 4.2: Complexity comparisons: Edge (C) and Edge (D) denote the edge sites for EP-BRVFL-AE(C) and EP-BRVFL-AE(D) respectively.

anomaly scoring methodologies on EP-BRVFL-AE(C) in Table 4.3, the predictive variance score is the most consistent as the number of sites increase. RE can be used for further inspection of $f(\hat{\mathbf{x}})$ against $\hat{\mathbf{x}}$ to identify the type of anomaly by breaking it down to individual attributes. Also, it can be seen that $\mathcal{H}$ is dominated by RE.

	UNSW-NB15			**Shuttle**		
No. Sites:	**500**	**100**	**1**	**500**	**100**	**1**
RE	89.88	89.91	89.93	86.67	94.19	**99.74**
$\phi(\hat{\mathbf{x}})^2$	**89.99**	**89.99**	**89.98**	**95.30**	**96.64**	99.39
$\mathcal{H}$	89.88	89.91	89.93	86.67	94.19	99.74

Table 4.3: AUC of different scoring methodologies on EP-BRVFL-AE(C) for different number of sites on UNSW-NB15 and Shuttle dataset.

Figure 4.3 shows AUC against different multiplicative factors ζ on the nodes in the hidden layer. In general, the more nodes in the middle layer, the better results but it comes at a higher computational cost with a larger B. The results show that a factor of $\zeta = 5$ is sufficient.

Results of experimenting with a small number of data points in each

Data points	UNSW-NB15		Shuttle		Aus Credit	
at each site:	5	10	5	10	5	10
$\phi(\hat{\mathbf{x}})^2$	72.45	75.88	94.72	95.09	76.04	87.84
$\mathcal{H}$	70.60	74.04	95.54	95.22	77.70	87.65

Table 4.4: AUC of EP-BRVFL-AE(C) with small number of data points at each of the 10 sites.

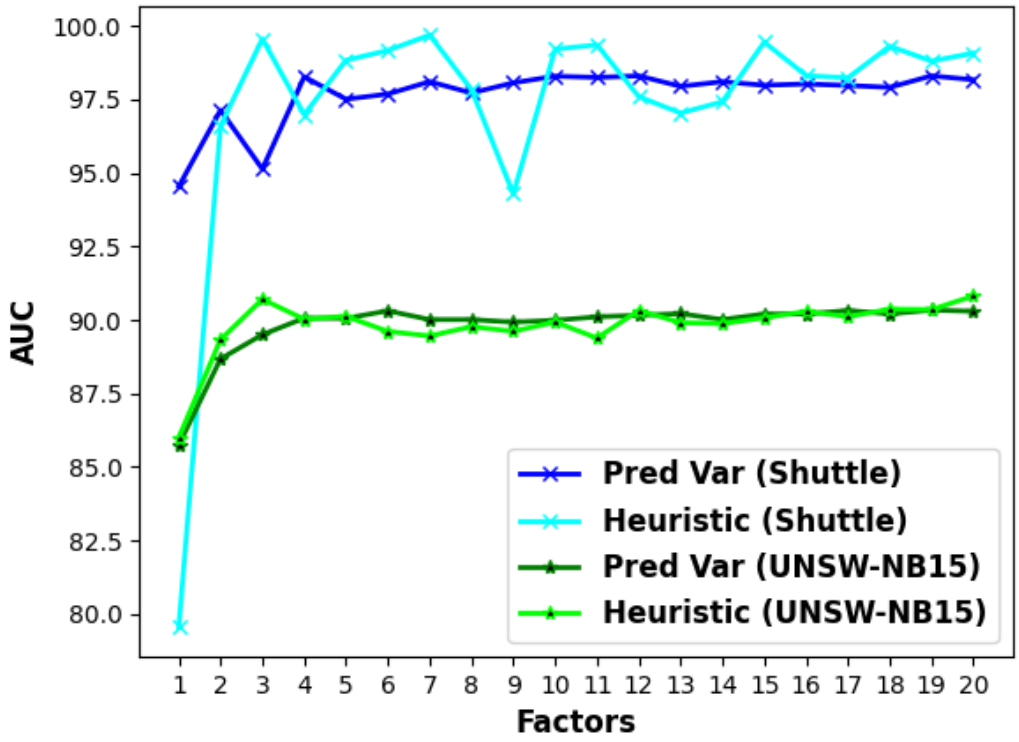


Figure 4.3: AUC against factor ζ for EP-BRVFL-AE(C) on the Shuttle and UNSW-NB15 datasets.

site are shown in Table 4.4. The data points are sampled at random. Though the EP-BRVFL-AE(C) still performs with few data points, in this case, it is more beneficial to send the data to a central site as $B > n_k$ unless there are privacy concerns.

The EP-BRVFL-AE(C) is compared with various other approaches, namely

PCA, k-NN, LOF, GMM, Bayesian GMM, OCSVM, RVFL-AE and AE. Each of these models are trained locally using data present in each site as done in [118, 147]. The results over 10 and 50 sites are reported in Table 4.5. The same test set is used across all sites and the average AUC and its standard deviation is reported. In the centralised approach, all data are sent to the central site where the model is trained.

Parameters for these other approaches are determined by commonly used rules of thumb. For PCA, the number of principal components (and for the AE the number of nodes in the hidden layer) is $\sqrt{d} + 1$ [27]. The number of nearest-neighbours for KNN and LOF is $\sqrt{n_k}$. The GMM is trained using expectation maximisation [20]. The Bayesian GMM is trained using VI and Dirichlet process weight concentration [21]. The BRVFL-AE is trained locally and parameters are not shared using EP. Weights for the RVFL-AE are determined using (4.4). For BRVFL-AE and EP-BRVFL-AE, if the heuristic measure $\mathcal{H}$ performs better, it is reported in brackets; otherwise, the predictive variance measure is reported. These results are shown in Table 4.5.

From Table 4.5, the most consistent performing model over any number of sites is the EP-BRVFL-AE. The difference between the EP-BRVFL-AE and BRVFL-AE at the central site is due to the hyper-parameter optimisation. There is no standard deviation on EP-BRVFL-AE because the global parameters are shared across all sites. Sharing parameters using EP also improves the AUC result on the Australian Credit and Abalone datasets. Ranking the performance, the top three performing methods are EP-BRVFL-AE (C), BRVFL-AE and Bayesian GMM, which suggests that the Bayesian approaches are best for distributed training. As for the other methods, there is no clear consistent model. In some cases the standard deviation increases with number of sites, which shows that models trained only on local data can have different results. The Bayesian GMM and GMM show good overall performance but fail when there is not enough data at the local site, as can be seen with the Australian Credit dataset.

Datasets	**UNSW-NB15**			**NSLKDD**		
	Local		**1**	**Local**		**1**
No. Sites:	**50**	**10**		**50**	**10**	
EP-BRVFL-AE(C)	**89.98**	**89.98**	**89.98**	**96.09**	96.09	96.09
BRVFL-AE	88.91 ± 1.02	89.81 ± 0.29	89.98	96.05 ± 0.17	96.16 ± 0.08	96.09
PCA	74.09 ± 1.69	73.75 ± 0.61	73.78	95.52 ± 0.25	95.50 ± 0.14	95.51
KNN	81.45 ± 0.83	83.74 ± 0.26	86.12	95.26 ± 0.18	94.92 ± 0.21	94.33
LOF	66.16 ± 3.38	81.32 ± 0.79	88.86	85.20 ± 2.81	83.91 ± 0.82	87.30
Bayesian GMM	88.77 ± 2.57	88.33 ± 2.48	87.29	95.59 ± 0.39	95.47 ± 0.40	95.25
GMM	83.70 ± 2.70	82.31 ± 0.77	84.62	95.45 ± 0.93	**96.17** ± 0.87	**96.59**
OCSVM	79.84 ± 1.06	79.94 ± 0.35	79.96	93.60 ± 0.15	93.65 ± 0.06	93.65
RVFL-AE	87.86 ± 0.85	88.73 ± 0.29	89.81	94.98 ± 0.21	95.21 ± 0.12	95.75
AE	81.19 ± 0.19	81.21 ± 0.08	81.21	91.69 ± 0.12	91.68 ± 0.06	91.68
Datasets	**Shuttle**			**Abalone**		
	Local		**1**	**Local**		**1**
No. Sites:	**50**	**10**		**50**	**10**	
EP-BRVFL-AE(C)	97.17	98.28	99.39	**75.70**	75.09	74.50
BRVFL-AE	95.39 ± 0.51	95.75 ± 0.51	97.26	72.74 ± 4.97	74.56 ± 1.87	74.62
PCA	84.95 ± 3.56	83.91 ± 3.35	83.14	68.12 ± 6.42	66.94 ± 2.83	65.73
KNN	97.92 ± 0.56	97.34 ± 0.62	98.05	59.32 ± 5.63	71.11 ± 3.04	77.98
LOF	98.34 ± 0.71	93.38 ± 2.30	98.79	51.87 ± 6.56	66.64 ± 2.86	74.15
Bayesian GMM	99.63 ± 0.11	99.23 ± 1.23	99.70	74.30 ± 4.49	**78.67** ± 1.07	**81.84**
GMM	**99.76** ± 0.14	**99.86** ± 0.02	**99.90**	60.10 ± 8.27	72.06 ± 2.65	81.38
OCSVM	96.51 ± 0.55	96.50 ± 0.23	96.49	55.02 ± 4.27	55.52 ± 2.01	55.78
RVFL-AE	94.33 ± 0.34	95.23 ± 0.16	95.01	64.52 ± 4.48	76.35 ± 1.78	76.69
AE	91.53 ± 0.19	91.53 ± 0.07	91.53	37.54 ± 3.49	36.37 ± 1.07	35.88
Datasets	**Australian Credit Approval**			**PageBlocks**		
	Local		**1**	**Local**		**1**
No. Sites:	**50**	**10**		**50**	**10**	
EP-BRVFL-AE(C)	**85.57**	**83.67**	80.82	**97.34**	**97.44**	**97.56**
BRVFL-AE	69.32(72.54)±10.44	73.44 ± 7.24	79.86	96.35 ± 0.64	97.05 ± 0.33	97.49
PCA	72.80 ± 10.93	71.31 ± 5.46	71.43	95.04 ± 1.18	95.24 ± 0.63	95.34
KNN	75.15 ± 9.24	81.13 ± 4.71	85.33	95.04 ± 0.85	95.78 ± 0.25	96.36
LOF	71.98 ± 12.06	79.05 ± 5.40	81.82	96.22 ± 1.68	93.92 ± 1.04	95.06
Bayesian GMM	50 ± 0.00	68.30 ± 5.71	82.08	96.04 ± 0.84	95.94 ± 0.53	96.49
GMM	49.99 ± 0.07	50.00 ± 0.22	78.02	92.89 ± 2.52	93.58 ± 2.05	94.19
OCSVM	77.95 ± 8.94	83.51 ± 3.66	**86.84**	96.25 ± 0.73	96.51 ± 0.24	96.59
RVFL-AE	74.85 ± 9.80	71.22 ± 5.57	86.80	97.02 ± 0.37	97.18 ± 0.19	95.38
AE	77.33 ± 9.18	83.32 ± 2.59	85.07	89.72 ± 1.46	89.62 ± 0.68	89.54

Table 4.5: AUC over various datasets and methods. AUC using $\mathcal{H}$ is given in brackets if it performs better than $\phi(\hat{\mathbf{x}})^2$ for bayesian implementation. The data are randomly distributed. Both mean and standard deviation (mean ± standard deviation) is reported for methods where the model is learnt using local data at each site and results are averaged.

To compare to other distributed methods such as MVE-PCA [147], do-OCSVM and sparse doOCSVM [118], EP is performed on the same data sets used in those studies. From Table 4.6, the EP-BRVFL-AE(C) performs comparably to other methods in the literature.

Table 4.6: Comparisons of AUC of different models

Datasets	**Central**		**20 sites**	
	MVE-PCA	EP-BRVFL -AE(C)	MVE-PCA	EP-BRVFL -AE(C)
Abalone	**83.28**	74.5	**82.73**	75.35
Shuttle	98.41	**99.66**	94.68	**97.83**
	Central		**10 sites**	
Aus Credit	80.77	**80.82**	73.98	**83.67**

Datasets	**50 sites**		
	doOCSVM	Sparse doOCSVM	EP-BRVFL-AE(C)
Abalone	63.41	64.52	**75.70**
PageBlocks	94.71	95.28	**97.38**

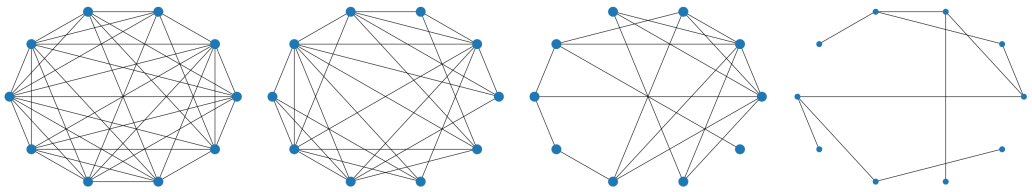
Fully Distributed

EP-BRVFL-AE(C) gives the same result as EP-BRVFL-AE(D) when $\kappa = 1$. Table 4.7 gives the results with different number of sites and κ values. The network configurations are implemented at random for the various κ values. The one with 10 sites is shown in Figures 4.4 and 4.5 shows the configuration with 10 and 50 sites respectively. For each network, the method is run for its Max Hops+2 iterations. The small standard deviation in AUC scores suggest that global solution at each site is almost similar to having a central site gathering all updates. This is further affirmed by the small average relative difference values, $E_{rel}(\mathbf{r})$ and $E_{rel}(\mathbf{Q})$. The results

show that the method works irrespective of κ. Similar results are observed on the other datasets.

Network				Convergence		AUC (Mean $\pm$ Standard Deviation)	
Sites	κ	**ADPS**	**Max Hops**	$E_{rel}(\mathbf{r})$	$E_{rel}(\mathbf{Q})$	$\phi(\hat{\mathbf{x}})^2$	$\mathcal{H}$
10	0.8	7.2	2	0.0775	0.0029	89.98 $\pm$ 0.000	89.93 $\pm$ 0.000
10	0.6	5.4	3	0.0448	0.0017	89.99 $\pm$ 0.000	89.93 $\pm$ 0.000
10	0.4	3.6	3	0.0607	0.0023	89.99 $\pm$ 0.000	89.93 $\pm$ 0.000
10	0.222	1.11	6	0.0182	0.0006	89.99 $\pm$ 0.000	89.93 $\pm$ 0.000
50	0.8	39.2	2	0.1691	0.0064	89.98 $\pm$ 0.000	89.92 $\pm$ 0.000
50	0.4	19.6	3	0.1228	0.0046	89.98 $\pm$ 0.000	89.92 $\pm$ 0.000
50	0.2	9.8	4	0.0875	0.0033	89.98 $\pm$ 0.000	89.98 $\pm$ 0.000

Table 4.7: Performance of EP-BRVFL-AE(D) on UNSW-NB15 dataset. Average values over the sites on the last iteration are reported for $E_{rel}(\mathbf{r})$ and $E_{rel}(\mathbf{Q})$ against the solution for EP-BRVFL-AE(C). Mean and standard deviation of Area under ROC curves (AUC) using both $\phi(\hat{\mathbf{x}})$ and $\mathcal{H}$ over the distributed sites are reported.

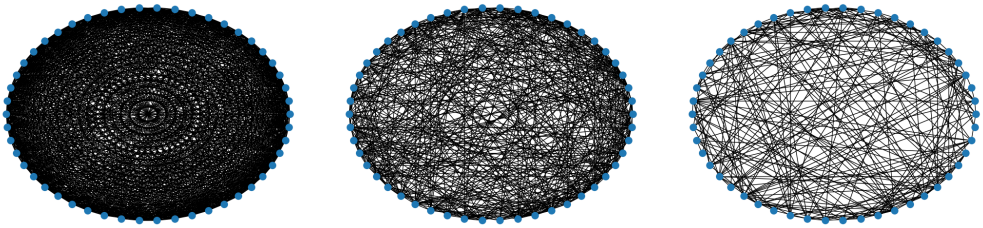


(a) $\kappa = 0.8$ (b) $\kappa = 0.6$ (c) $\kappa = 0.4$ (d) $\kappa = 0.222$

Figure 4.4: Networks with 10 sites

Biased Partitions of Data

In some networks, distribution of data from individual sites may be different. Furthermore, the number of data points could vary widely. The worst-case scenario is where data in each site have different profiles. Using a GMM, the data is split into 10 such that each site contains data from



(a) $\kappa = 0.8$ (b) $\kappa = 0.4$ (c) $\kappa = 0.2$

Figure 4.5: Networks with 50 sites

Network				**Convergence**		**AUC** (Mean $\pm$ Standard Deviation)	
Sites	κ	**ADPS**	**Max Hops**	$E_{rel}(\mathbf{r})$	$E_{rel}(\mathbf{Q})$	$\phi(\hat{\mathbf{x}})^2$	$\mathcal{H}$
10	1	9	1	0.0023	0.000	89.76 $\pm$ 0.000	89.46 $\pm$ 0.000
10	0.8	7.2	2	0.0758	0.0029	89.75 $\pm$ 0.162	89.40 $\pm$ 0.141
10	0.6	5.4	3	0.2419	0.0084	89.55 $\pm$ 0.200	89.34 $\pm$ 0.137
10	0.4	3.6	3	0.2668	0.0091	89.44 $\pm$ 0.674	89.19 $\pm$ 0.604
10	0.222	2.0	6	0.3406	0.0080	89.57 $\pm$ 0.616	89.78 $\pm$ 0.683

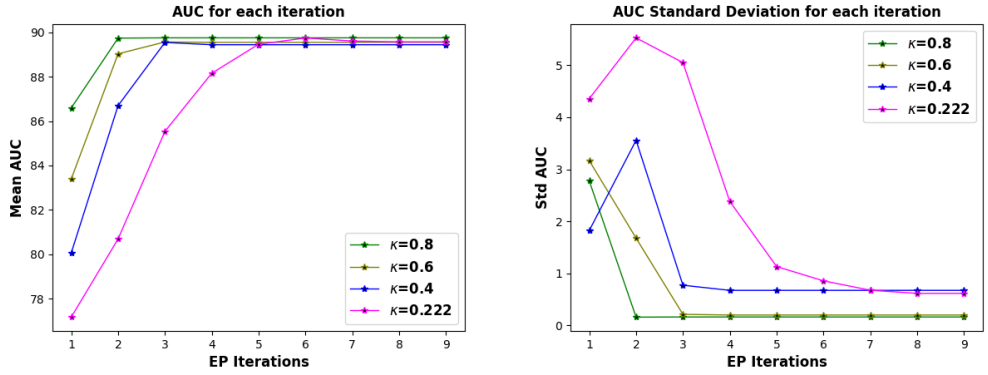
Table 4.8: Performance of EP-BRVFL-AE(D) on UNSW-NB15 with biased and uneven partitions of data.

a separate Gaussian component. The UNSW-NB15 dataset is used and the results are found in Table 4.8.

Figure 4.6a shows mean AUC increasing and Figure 4.6b shows standard deviation amongst AUC over all sites decreasing as updates are being received at each iteration. Figures 4.6c and 4.6d show relative errors decreasing with respect to EP-BRVFL-AE(C). The results show that at the iteration after the Max Hops of the network, the solutions converge. This also further validates that the first update from each site is the most important. Hence, s in the computational complexity is at most the Max Hops value. From Table 4.8 and Figure 4.6, the solutions for the worst case scenario on the poorest distributed network for 10 sites of $\kappa = 0.222$ still converge. This implies that the solution will converge for any network configurations.

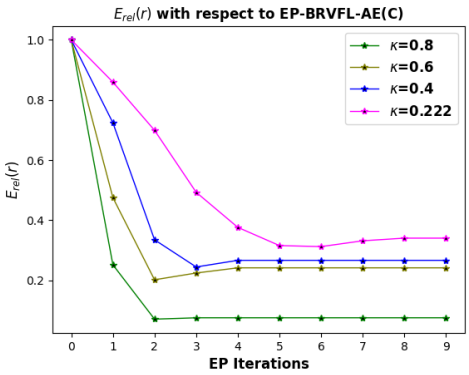
Furthermore, EP-BRVFL-AE(C) achieves an AUC of 89.76 and 89.46 for $\phi(\hat{\mathbf{x}})^2$ and $\mathcal{H}$ measures respectively with GMM split. An important observation is that EP-BRVFL-AE(D) with $\kappa = 1$ achieves the same result in Table 4.8, despite the biased and uneven partition of data. This implies that updates from each site can still be combined using EP to build the model. Hence, logically, the model is robust under transmission delays. The computation at each site can continue and the update can be included in the following iteration. The worst case scenario is simulated where updates from each site arrive one at a time and the computation for the global parameters is performed after each arrival. Figure 4.7a shows how the global parameters converge to the scenario where all updates arrive together at the central site. The AUC score increases as each update from the site is included as depicted in Figure 4.7b. Hence, asynchronous updates are possible with the information received at each site.

The value of δ remains close to the initial value of 1 when the data is evenly distributed. For the case of GMM split, Figure 4.8 shows the minimum and average δ values for $\mathbf{Q}_{-k}$ to remain positive definite at different κ values. Most of the time, as δ remains unchanged, $t = 1$ in the computa-

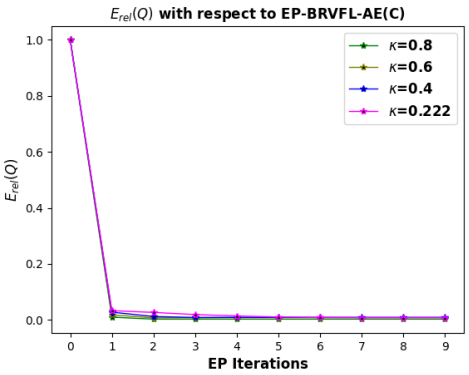


(a) AUC

(b) Standard Deviation of AUC

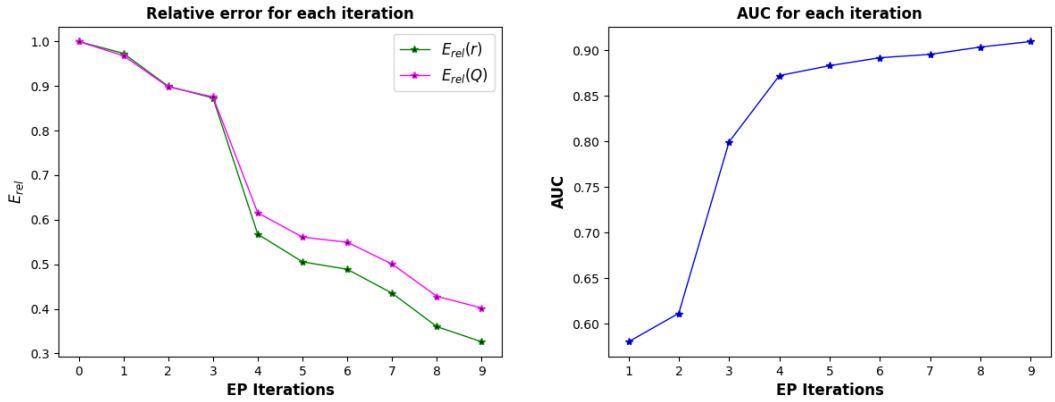


(c) $E_{rel}(\mathbf{r})$



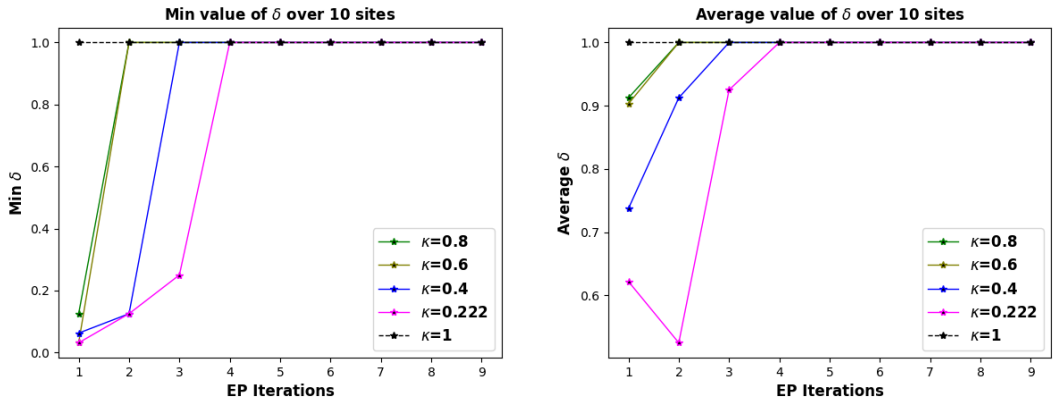
(d) $E_{rel}(\mathbf{Q})$

Figure 4.6: EP-BRVFL-AE(D) with 10 sites evaluated at each EP iteration on the UNSW-NB15 dataset with GMM split.



(a) $E_{rel}(\mathbf{r})$ and $E_{rel}(\mathbf{Q})$ (b) AUC

Figure 4.7: EP-BRVFL-AE(C) with 10 sites where updates are combined one by one on UNSW-NB15 with GMM split.



(a) Minimum δ (b) Average δ

Figure 4.8: (a) Minimum and (b) Average of δ over 10 sites during each EP iteration

tional complexity.

4.3 EP-BAE

EP-BRVFL-AE gives a good model for distributed training and anomaly detection. The SLFN provides a fast, efficient and simple solution to work at the edge of the network. One of the essentials it lacks is depth in the neural network and dimensionality reduction that is achieved by the offline model, the AE in Chapter 3. Depth of a neural network is important to model complicated data such as in image recognition. This section explores whether a deep Bayesian AutoEncoder (BAE) can replace the BRVFL-AE.

One immediate implication is that with more hidden layers in the neural network with non-linear activations, Bayesian inference is no longer analytically tractable. Hence a closed form solution does not exist. In algorithms 3 and 4, the computation of the tilted distribution needs to be replaced by approximation methods such as VI [20]. Furthermore, the immense computation required would stress edge devices.

4.3.1 Performance

Model	VI	BOHAMIANN
BAE with data at central	74.07	90.84
EP-BAE (C) with 5 sites	76.54	74.16

Table 4.9: AUC scores on UNSW-NB15 ML dataset

An AE with 3 hidden layers is used. The AUC scores of BAE trained with all data in one location and EP-BAE with data at 5 sites are shown under VI and BOHAMIANN in Table 4.9. ADVI [99] was used to compute the tilted distribution for the BAE. The MAP estimate of the posterior is used as a one value estimate of the weights and the RE is used as

the anomaly measure. VI is approximating the posterior to a product of independent Gaussian distributions. There are other methods to train a BNN such as BOHAMMIAN [177], Bayes-By-Backprop [23], Probabilistic Backpropagation [75], Dropout [61] and Stochastic Gradient Hamiltonian Monte-Carlo (SGHMC) [34], amongst which BOHAMMIAN shows superior performance [177]. BOHAMMIAN is essentially performing MCMC sampling from the posterior and works well with all of the data at the central as shown in Table 4.9. This is because BOHAMIAN samples from the true posterior which is unlikely to be Gaussian. However, this poses a problem for EP because the posterior at each site needs to be mapped to a member of the exponential family of distributions. This process will affect the accuracy a great deal. For example, if the posterior is bimodal as in Figure 4.9, a map to a Gaussian distribution would put the mean in the middle where it is furthest from the mode. Using VI to approximate the posterior, allows to obtain one of the modes as a Gaussian. And this is a better approximation to be fed into the EP iterations for distributed training. Thus, it performs more consistently. However, convergence might also take longer if the distribution is projected onto Gaussians with different modes during the EP iterations.

Next the EP-BAE which is now trained in a distributed manner, is combined with the two best performing online models from Section 3.2.The results are shown in Tables 4.10 and 4.11. In this scenario the IOCSVM does not perform as well, this is because it is difficult to train the EP-BAE to be a strong classifier of existing data (i.e. overfit). Incorrect support vectors hugely influence IOCSVM. The IGMM method does not face this issue and it improves the offline EP-BAE. Table 4.10 gives the result of using EP-BAE(D) as the offline model.

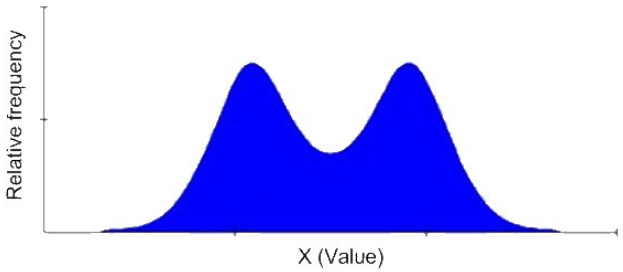


Figure 4.9: An example Bimodal distribution

Model	UNSW-NB15	
	AUC	AP
EP-BAE(D)	96.71	62.79
EP-BAE(D)+IOCSVM	92.61	46.60
EP-BAE(D)+IGMM	98.43	85.96

Table 4.10: Average AUC and AP score of EP-BAE(D) on streaming datasets over 5 sites

4.3.2 System Implementation

A block diagram for EP-BRVFL-AE(D) and EP-BAE(D) at the edge site is depicted in Figure 4.10. After convergence, the data used to train the anomaly detection model can be discarded to save memory. The method can be adapted to perform online batch training with new data as well. It was shown in Section 4.2.5 that the optimal waiting time is equivalent to

Model	UNSW-NB15	
	AUC	AP
EP-BAE(D)	95.85	74.38
EP-BAE(D)+IOCSVM	93.40	61.62
EP-BAE(D)+IGMM	97.43	82.73

Table 4.11: Average AUC and AP score of EP-BAE(D) on streaming datasets with GMM split on 5 sites

the max hop time for any two sites in the network to communicate their updates.

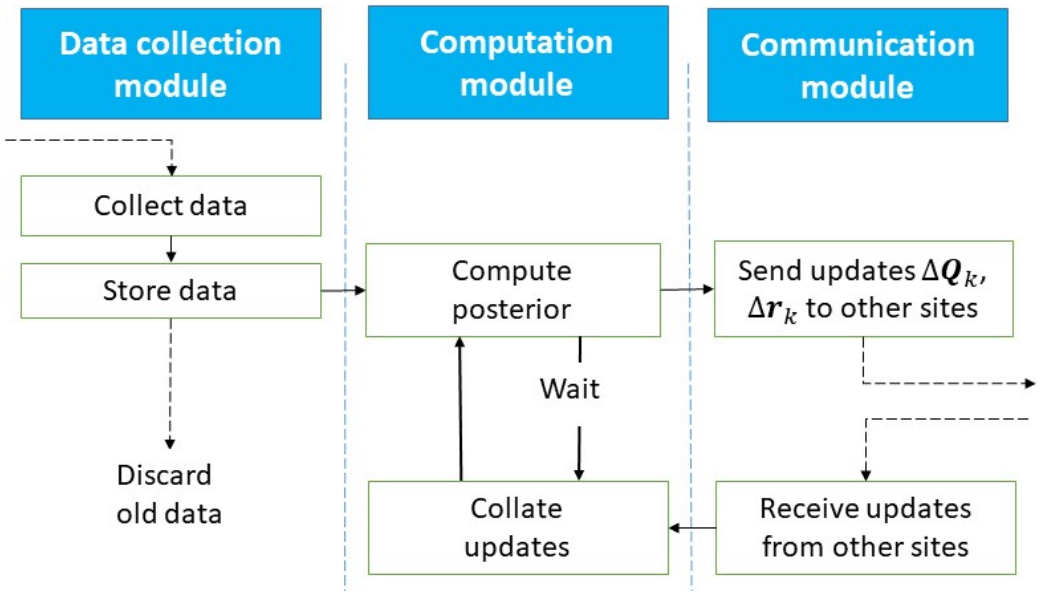


Figure 4.10: Block Diagram of EP-BRVFL-AE(D) implementation on edge site

4.4 Conclusion

In this chapter, a novel EP-BRVFL-AE for anomaly detection in Edge AI networks is described. The model is trained in a distributed manner without having to share data from each site. Only changes in the posterior parameters of the EP-BRVFL-AE weights are shared instead of raw data. Furthermore, the use of the conjugate prior and SLFN ensues a closed-form solution which achieves rapid convergence. The longest wait time is the number of Max Hops within a network. Moreover, asynchronous update is also possible.

The EP-BRVFL-AE was evaluated against other methods in the literature and the results were comparable. It was also evaluated in the dis-

tributed setting with varying network densities and has been shown to be stable in terms of performance. Worst case scenario analysis was performed by splitting the data in a biased and uneven manner and the model was able to converge. The algorithm performs well in general under all of the mentioned scenarios. The method of using EP has been extended to train a BAE as the offline model and the Hybrid Online Offline Framework has been shown to improve on the purely offline model. However, this incurs higher computational complexity and requires a longer time to achieve convergence.

Thus far evolving and heterogeneously distributed data have been considered for anomaly detection and the methods have been validated on available public datasets. However, these datasets are available in formats suitable for machine learning models and it is not always the case in the real world. It is rare that raw data can be directly applied to machine learning models. In the following chapter, three different generic real world domains are studied and each poses a unique challenge. The methods developed in Chapters 3 and 4 are further validated on a scenario in each domain.

Chapter 5

Real World Scenarios

The literature on anomaly detection focuses heavily on methods but less on the many challenges faced in the real world. This chapter addresses the unique challenges involved in effectively implementing an anomaly detection model in three generic domains, namely systems monitoring [140], network graphs monitoring and surveillance monitoring.

5.1 Systems monitoring

Industrial equipment monitoring, data center operations monitoring and power station monitoring are some examples of systems where the performance need to be monitored regularly. When a fault occurs, it needs to be detected and the cause identified quickly. To build a robust anomaly detection system in this domain, the informative variables must be selected. This section studies the challenges of systems monitoring domain applied to a rural ISP: Venture Networks, located in the Horowhenua District of New Zealand.

5.1.1 Introduction

To provide Internet connectivity to rural communities, ISPs need to deploy and maintain many wireless sites in isolated or inaccessible terrain. Addressing failures at such sites can be very expensive, both in identifying the fault, and also in the repair or rectification. Data monitoring can be useful, to spot anomalies and predict a fault (and possibly pre-empt it altogether), or to locate and isolate it quickly once it causes an issue for the network. There might be hundreds of variables to be monitored in principle, but only a few of significance for detecting faults.

The ISP provides Internet connectivity to local farms, and has several sites across the region. Some of the sites are difficult to access, especially during the winter months, due to the challenging terrain and severe weather. In the last year, Venture Networks has been caught off guard by several unexpected failures at their sites. In one case, a battery bank failed before expected, and in another case, water penetrated a supposedly waterproof power junction box, causing half of the solar array to fail. When dealing with sites in challenging areas, often a trip to identify the fault will be required, which can be expensive, and worse, dangerous especially during adverse weather conditions.

In what follows, step-by-step methods are shown to set-up an anomaly detection model from scratch with a novel Bottom-Up approach to identify useful features, determine the cause of anomalies and update variables iteratively. The Hybrid Online Offline Framework is also validated in this scenario.

5.1.2 Methodology

ISPs may already subscribe to a third party data monitoring tool such as SolarWinds [175]. An ISP can also set up a monitoring tool from open source programs such as Zabbix [145]. In the former, an ISP has an abundant number of variables that can be monitored, and in the latter, an ISP

would need to build the variables up from scratch.

Even with an abundant number of variables, it is futile to indiscriminately throw all the variables into a ML algorithm in hope of learning some patterns. Not only will it be difficult, but patterns learnt cannot be easily interpreted. In the literature, there are many feature selection methods that reduce a set of variables to important ones [31]. This approach shall be referred to as the *Top-Down* approach. For this method, real world data that depicts anomalies is required and, being Top-Down, it may not provide enough context to enable proper insights into the faults. Thus, it is prudent to build the monitoring variables from scratch; in other words, from the knowledge of the network manager or the *Bottom-Up* approach. The Bottom-Up approach also allows managers to incorporate new variables that may be important but are not in the list of variables provided by third party monitoring solution providers, for example by including a sensor to monitor moisture or temperature near the batteries.

Bottom-Up Approach

Being aware of what variables are fed into the ML model is an important step which is usually ignored. Many practitioners input many variables and expect it to perform but putting in uninformative variables will not yield any useful results. Informative variables should come from knowledge of the system and the problem. To begin, the following questions were posed to the network manager.

1. Describe the nature of the faults or anomalies faced by the system.

2. To detect each of the above mentioned faults, which variables are currently monitored?

3. For each of the above mentioned variables, how are the data collected?

The rationale for the first question is to understand the faults that occur within the site. This can be battery faults, latency issues, radio frequency anomalies, etc. Each of these can be further broken down into subcategories. For instance, battery faults can be caused by either low-voltage, battery out of order or limited solar generation. It may be brought all the way to the individual battery or each link in the network. The objective is to make this list as fine-grained and as comprehensive as possible.

Next, from each of the sub-categories, variables which the network manager currently uses to detect the anomalies are identified. For example, high *Watt hours* and high *Current* can result in low voltage situations which affects performance as a whole. One of the difficulties in identifying these variables for ML is that network managers tend to understand the faults from a high level. "A low-voltage situation could be caused by excessive consumption or a lack of control of power generation." For ML methods, this cause needs to be further broken down to specific measurable variables by asking appropriate questions. Some examples include "How is consumption or power-generation measured?" This also helps the network manager think through their current processes and facilitates the building of a model of the fault that the data can point to, when it occurs.

Measurement

The next challenge is to accurately measure the identified variables. If the existing third party provider already measures and provides all of the required variables, then the task is simple. Otherwise, one would need to find a way to measure the variables directly or indirectly depending on the site architecture and set-up. For example, the 'ping' command can be used to measure RTT, weather information can be scraped off the web, and Signal-to-Noise ratio (SNR) measurements – as well as other spectrum analysis – can be performed using existing radios, or by setting up custom monitoring hardware within the environment.

Another challenge is to determine the appropriate time-window to make the measurements. This depends on the persistence of the site operation when a fault is detected. It also depends on the measuring capabilities. Some variables can only be measured every hour, while others can be measured every few seconds. Small intervals capture fine changes which may not be relevant in the context of site operation. It may also require more computation to measure, and additional storage space. Rendering an interval too large might miss pertinent information. Either way, it is necessary to study different time-windows during experimentation to gain specific insights.

Types of Variables

Certain variables are not numerical. For example the weather is a categorical variable and certain device information such as *state* of the device, *flags*, or *DIP switches* which only take a few numerical values should also be considered as categorical.

Some faults are detectable relative to previous values. This goes into second order calculation of variables such as *Change in Watt Hours* or 'Δ Watt Hrs' over two time-windows.

In a time-window, certain variables can be measured many times. For instance, *Battery Temperature* or *Charge Current* may be measured every 10 seconds. In a time-window of 5 minutes, one obtains 30 values. Statistical attributes such as mean, maximum, minimum, median, variance, and interquartile-range of variables can be used to describe the distribution of the 30 values. This depends on the variable and the anomaly of interest, i.e., which statistical attribute indicates the fault most accurately and is based on the knowledge and understanding of the network.

Table 5.1: Measured & Derived Variables

Variables	Description	Type	Derived Variables	Type
Battery Voltage	Instantaneous battery bank voltage (internal sensor)	Numerical	ΔBattery Voltage	2^{nd} Order
Battery SVoltage	Instantaneous battery bank voltage (external sensor)	Numerical	ΔBattery SVoltage	2^{nd} Order
Target Voltage	Target battery bank voltage	Numerical	–	–
Charge Current	Instantaneous charge controller charge current	Numerical	–	–
Output Power	Instantaneous power into the battery bank	Numerical	ΔOutput Power	2^{nd} Order
Input Power	Charge controller input power	Numerical	ΔInput Power	2^{nd} Order
Array Voltage	Instantaneous output voltage of solar array	Numerical	ΔArray Voltage	2^{nd} Order
Array Current	Instantaneous output current of solar array	Numerical	ΔArray Current	2^{nd} Order
Sweep Vmp	Maximum power voltage of the array	Numerical	–	–
Sweep Voc	Open circuit voltage of the array	Numerical	–	–
Sweep Pmax	Maximum power produced by the array	Numerical	–	–
Battery Temp	Battery temperature (external sensor)	Numerical	ΔBattery Temp	2^{nd} Order
Heat sink	Charge controller heat sink temp	Numerical	ΔHeat sink	2^{nd} Order
Amp Hours	Daily (moving) amp hours count	Numerical	–	–
Watt Hours	Daily (moving) watt hours count	Numerical	–	–
Weather Temp	Levin town temperature	Numerical	–	–
Weather Wind	Levin town wind speed	Numerical	–	–
Tx Capacity	Transmit capacity of the wireless link	Numerical	–	–
Rx Capacity	Receive capacity of the wireless link	Numerical	–	–
Signal	Received Signal Strength Indicator	Numerical	–	–
Noise Floor	Noise floor of the wireless link	Numerical	–	–
SNR	Signal-to-Noise Ratio of the wireless link	Numerical	–	–
RTT	Site-to-Site Round-trip Time	Numerical	–	–
Tx RTT	Radio transmit Round-trip Time	Numerical	–	–
Min Vb daily	Daily (moving) min battery voltage	Statistical	–	–
Max Vb daily	Daily (moving) max battery voltage	Statistical	–	–
Min Tb daily	Daily (moving) min battery temp	Statistical	–	–
Max Tb daily	Daily (moving) max battery temp	Statistical	–	–
Weather	Weather state (clouds/rain/sun, etc)	Categorical	–	–
Charge State	Current stage of the 4-stage charging algorithm	Categorical	–	–
DIP Switches	Hardware configuration switch state	Categorical	–	–
CCQ	Client Connection Quality	Categorical	–	–

5.1.3 Experiments and Results

Through the Bottom-Up approach, the following variables depicted in Table 5.1 were identified and measured. These are related to the main types of anomalies that have occurred frequently, namely power and latency.

These variables are a natural starting point to measure and input into ML methods. Through analysis and identification of faults, more variables can be identified and added later on as anomalies occur. For example, if a hardware fault is found to be caused by water damage, that cause could be incorporated by measuring moisture or water levels in the vicinity as a variable for future detection.

Various methods were used to measure the variables. Weather was recorded by polling a weather service Application Programming Interface (API), due to the lack of a local weather station. Most of the variables – except those related to weather, latency, or radio-specific functions – were captured using a commercially available solar charge controller. A time window of 5 minutes was used to capture each of the variables, and the results were stored on a server running within Venture Networks' central office.

As for preprocessing the data, categorical variables were converted to numerical values using one-hot encoding followed by PCA. Min-Max scaling was used to normalise the data. For this specific ISP, the data profile significantly varies during different hours of the day. Training one model over 24 hours would not suffice because if a situation that would usually occur during the night occurred during the day, that would be regarded as unexpected. Hence a new categorical variable is inserted to denote the hour interval.

An AE was trained by removing variables which the Network Administrator currently monitors, namely Battery Voltage, Battery SVoltage, ΔBattery Voltage and ΔBattery SVoltage. This is to test whether the anomalies can still be identified using other variables. In the same manner, further validation is done on the Hybrid model with the dataset.

With only the AE, Figure 5.1 shows how Battery Voltage changes during the course of the day. During normal operations, the anomaly scores are low. However, when there is an unexpected behaviour there are more spikes in anomaly scores and this has been determined solely based on other variables. This suggests that if Battery Voltage were not observed, the AE would still be able to detect the unexpected behaviour. Upon further analysis, the network manager can determine that Battery Voltage should be measured and be an input variable to the AE to make it more robust. The bottom graph of Figure 5.1 shows how one variable behaves. The top graph shows how all of the other variables move in correlation depicted on to one dimension to show whether it is moving normally or anomalously of a ML model and the benefits of using it to uncover hidden relationships.

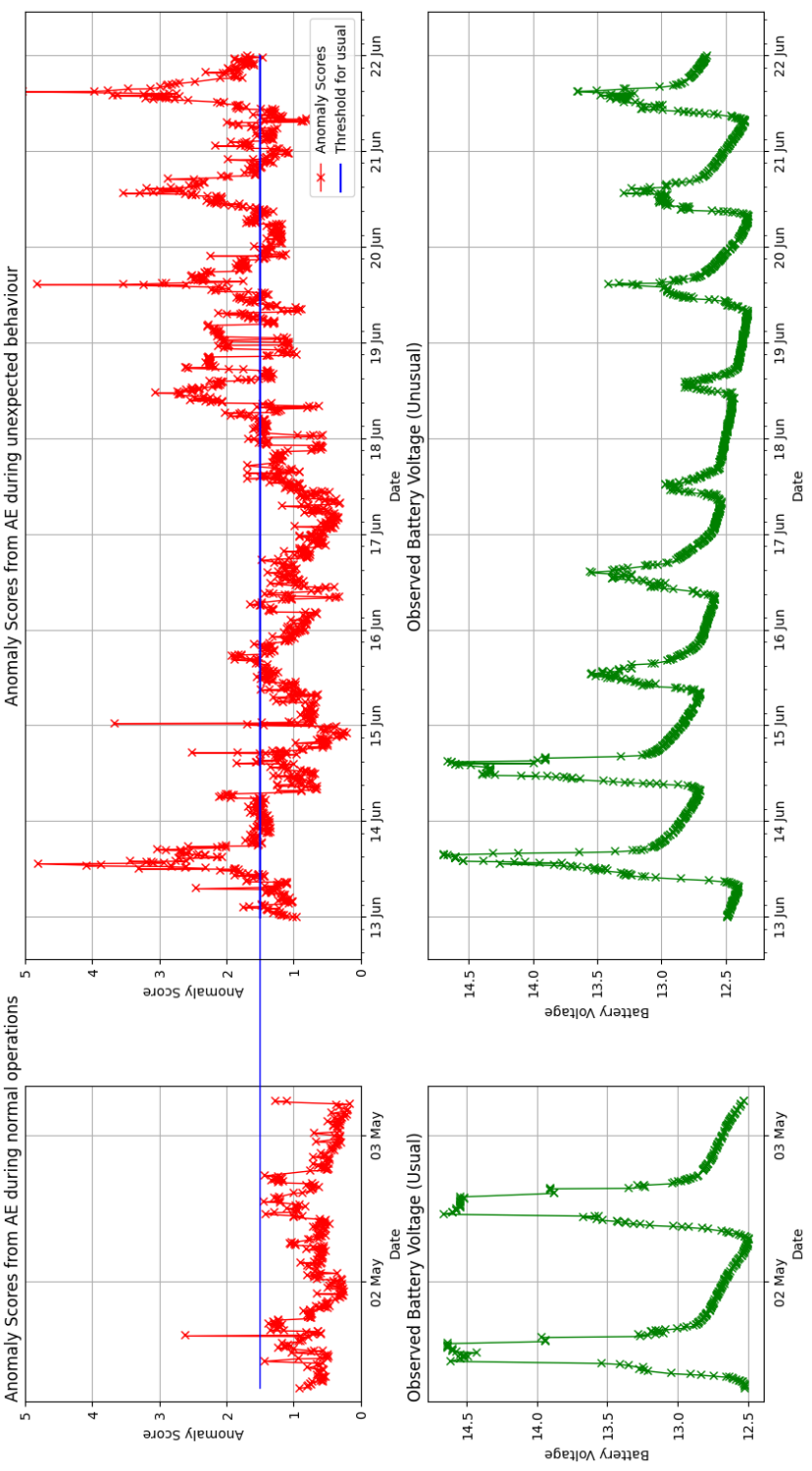


Figure 5.1: Left Bottom: Observed Battery Voltage variable during normal operations. Left Top: Anomaly scores from AE based on other variables during normal operations. Right Bottom: Observed Battery Voltage during some unexpected behaviour. Right Top: Anomaly scores based on other variables during unexpected behaviour.The threshold in blue is defined based on Left Top graph.

From the network manager's method of observing 'Battery Voltage', the problem was detected on the 17th of June after consecutive days of torrential rain.

The AE also detects more anomalies during that period and after. This is determined by comparing the anomaly scores during normal operations on left top of Figure 5.1. The threshold in blue is also determined in this manner. A few data points being above the threshold during normal operations is not surprising and could reflect measurement error. However, when many points are above the threshold, it can be concluded that there are some unexpected behaviours.

To determine the cause of the unexpected behaviour, feature identification was performed on the data during the anomalous period. The features are ranked based on χ^2 statistic [192] and Mutual Information (MI) [49] in Table 5.2. Both these methods make different assumptions about the data but regardless, it helps us to identify that the unexpected behaviour is due to the battery. Note that in the literature, these methods are used for feature identification before training a ML model [31]. Here it is done ***after*** identifying the anomaly to understand what type it is. This also helps us identify other important variables to monitor and the hidden relationships between them in describing the fault. For instance, 'Sweep Voc' should also be monitored instead of only 'Battery Voltage'. Other related variables to this battery anomaly are also learnt.

Since latency anomalies were not observed during the period, they were artificially injected by using the Linux utility traffic control. It ran on the server collecting anomaly data, so as to not degrade live customer connections. The results are shown in Figure 5.2. The results are shown in logarithmic scale since the anomaly scores were huge. The respective features are ranked in Table 5.3. However, by looking at the contribution of each feature to the anomaly score based on the values of the χ^2 statistic and Mutual Information (MI), only RTT stood out. This is because the latency anomaly was injected artificially instead of it having occurred in

real-time. Despite that, unexpected values in 'Input Power', 'Array Voltage' and 'Sweep Voc' are also be related to latency anomalies.

	Ranking based on	
Rank	χ^2 **statistic**	**Mutual Information**
1	Sweep Voc	Sweep Voc
2	Sweep Vmp	Min Vb daily
3	Array Voltage	Watt hours
4	Target Voltage	Amp hours
5	Charge Current	Max VB daily
6	Output Power	Max TB daily
7	Input Power	Weather Temp

Table 5.2: Features that describe unexpected behaviour in Figure 5.1

	Ranking based on	
Rank	χ^2 **statistic**	**Mutual Information**
1	RTT	RTT
2	Target Voltage	Sweep Voc
3	Sweep Vmp	ΔInput Power
4	Array Voltage	Array Voltage
5	Sweep Voc	Input Power
6	Output Power	Weather Temp
7	Input Power	Sweep Pmax

Table 5.3: Features that describe latency in Figure 5.2

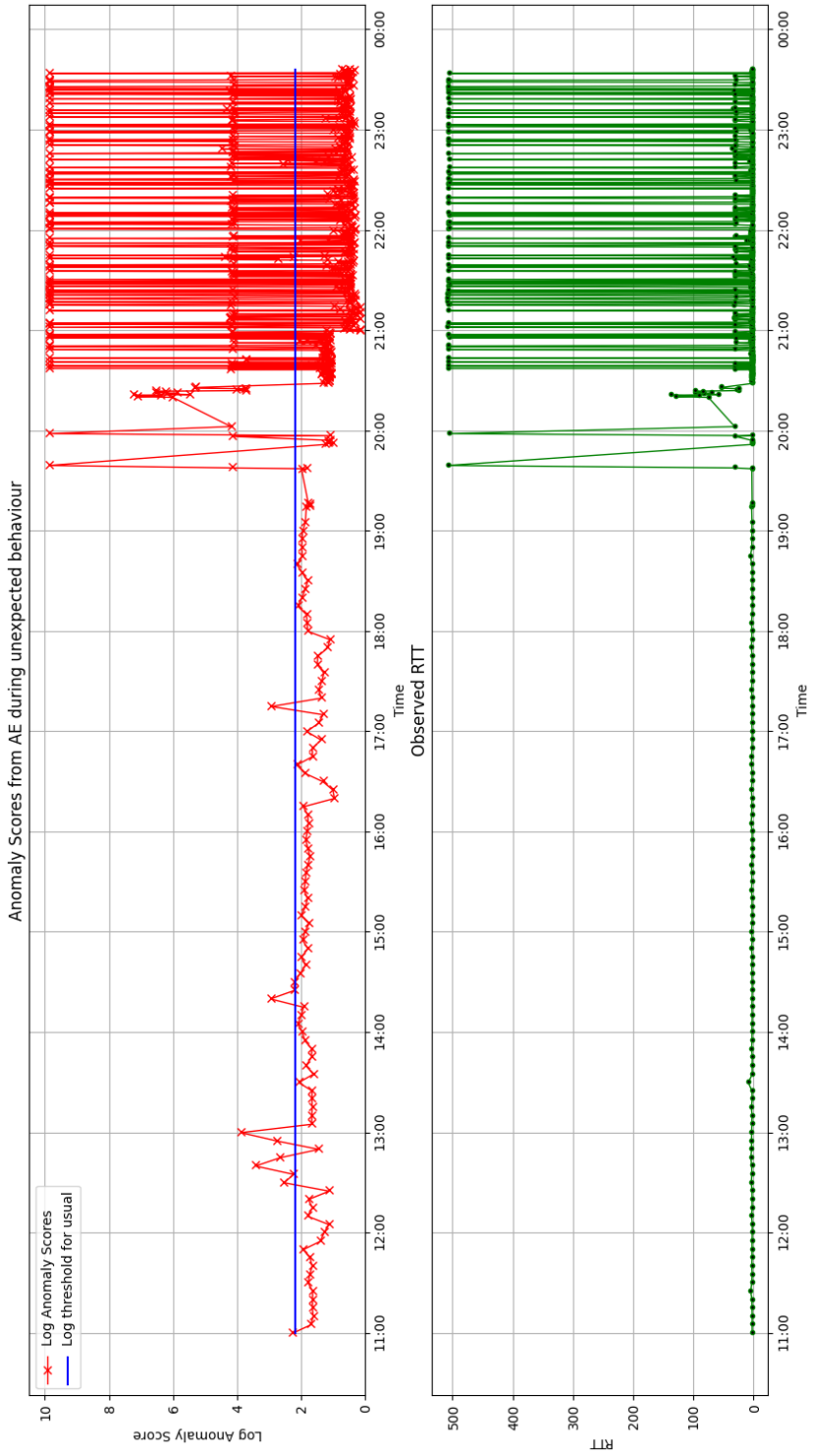


Figure 5.2: Top: Log anomaly scores from AE based on other variables during normal operations and during injected latency after 1930. Bottom: RTT. The threshold in blue is the same value as in Figure 5.1. All log anomaly scores are translated to above 0.

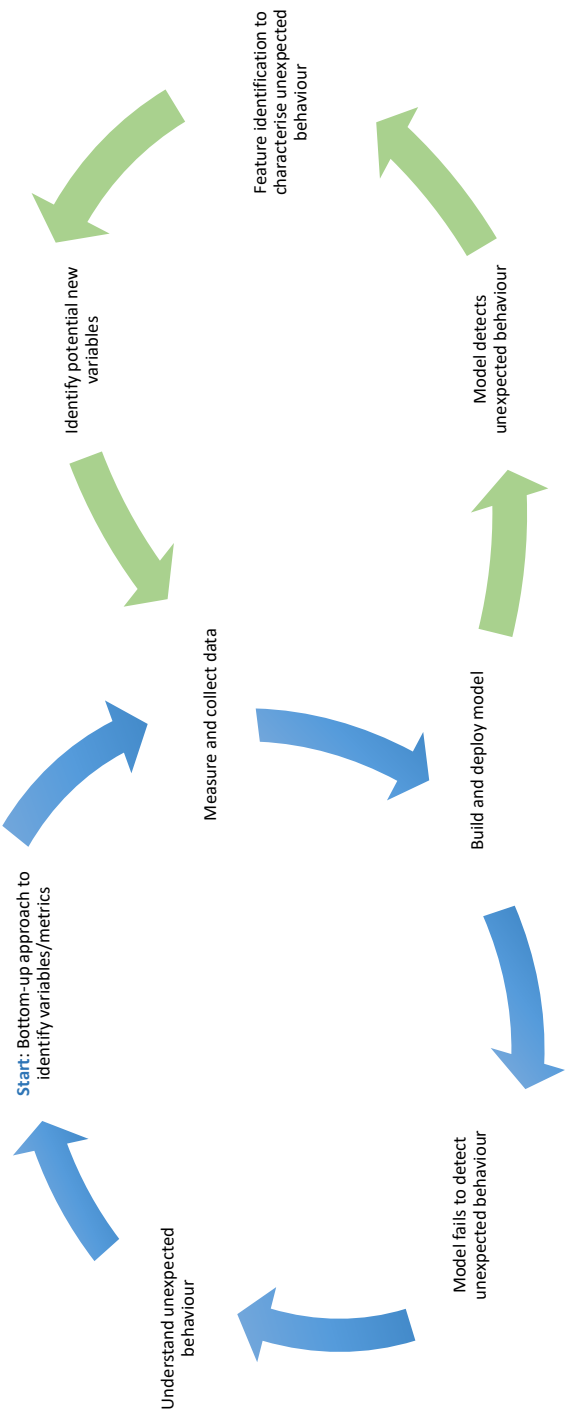


Figure 5.3: An iterative procedure to identify variables and build the model. After a couple of iterations, the procedure will reach the green loop where the model detects all related anomalies for the network manager.

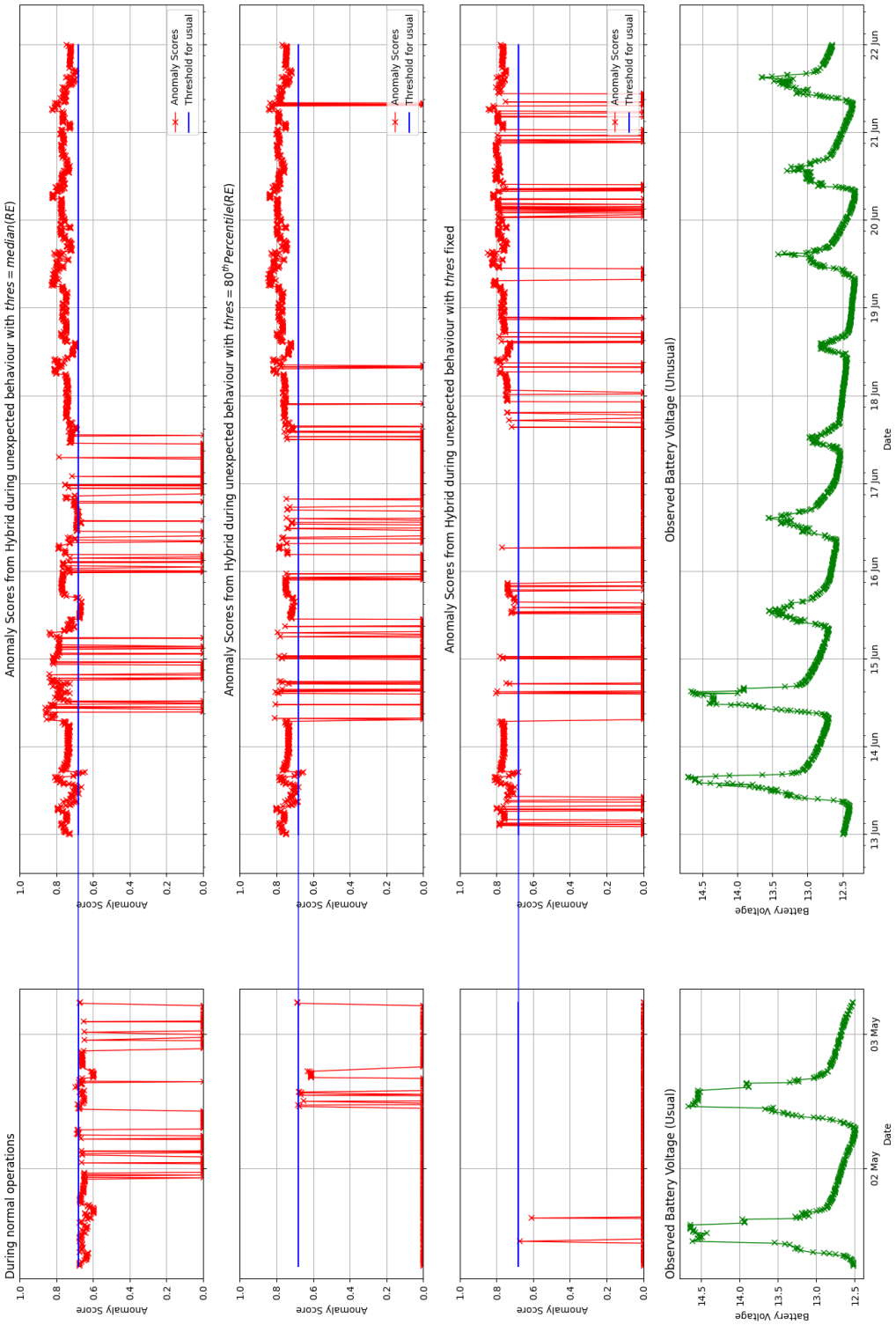


Figure 5.4: Top, second, third: Anomaly scores from AE+MW+IOCSVM based on other variables during normal operations (left) and unexpected behaviour (right) with $thres = median(RE)$, $thres = 80^{th}Percentile(RE)$ and $thres$ fixed respectively. Bottom: Observed Battery Voltage variable during normal operations (left) and unexpected behaviour (right).

Hence from the above results, obtaining the right variables is pertinent for a proper anomaly detection system. A hypothesis that certain variables can help to detect anomalies can be a good starting point. This is the Bottom-Up approach described in section 5.1.2. Subsequently, as anomalies are detected, feature identification can be performed to categorise the fault and to find important relationships between the variables that describe the fault. The network manager can identify potential new variables to measure for fine grained results in future for this specific unexpected behaviour. However, it is impossible to encompass all possible anomalies at the start. If an anomalous behaviour is missed, which the network manager has been made aware of from other sources such as customer complaints, it means that the model has not been provided with the right variables to detect that particular anomaly. Hence, this is an iterative process depicted in Figure 5.3 to identify variables over time and continually build a robust model. The next time that particular anomaly occurs, it can be easily detected and mitigated.

This process also helps the network manager understand exactly how the network operates, identify the specific cause(s) of a fault and address it instead of using 'patchwork' methods such as boosting the signal or increasing power. After a robust model is built, this also allows for automation of network operations.

The Hybrid AE+MW+IOCSVM model from Chapter 3 is tested on this dataset and the results are shown in Figure 5.4. Compared to Figure 5.1, overall, it can be seen that the hybrid model provides better discrepancy between anomalous points and normal points. This goes to validate that the hybrid model performs better than using only the offline model on this real world dataset. Three graphs are presented by adjusting the $thres$ variable in Figure 3.1 and Algorithm 1. In the top graph, the conservative approach of median for $thres$ is taken. This works but during normal operations, most of the anomaly scores are also close to the threshold (blue line in Figure 5.4). In the second graph, $thres = 80^{th} Percentile(RE)$ is

taken and this shows a good balance between detecting anomalous days and normal days. With the fixed $thres$, there are more false negatives on the anomalous days and this could result in missing the anomaly when observing the scores live.

In the subsequent section, the challenges in Network Graph monitoring are studied in similar fashion.

5.2 Monitoring Network Graphs

From the beginning of the Internet there have been connected devices such as routers transferring information. These connections are constantly changing with updates or when devices are added or removed. However, sometimes the changes are malicious [5, 10]. This section studies the challenges of capturing network connection information using Network graphs applied to BGP updates.

5.2.1 Introduction

The global Internet is connected using core routers which route traffic between users and servers. The routers use connectivity information such as IP addresses to determine where to direct network traffic. The BGP [155] prescribes the connections between Autonomous Systems (ASs) and it forms the backbone of the global Internet. However in recent times, this protocol has been susceptible to hijacking [5]. Hijacking is a form of application-layer DDoS attack which allows an attacker to impersonate a network using a legitimate network prefix as their own. This allows traffic to be inadvertently forwarded to the attacker if the attempt is not detected quickly. Over the last decade, there have been many such BGP hijacking incidents [5]. Some examples include loss of connectivity for domains of an ISP "panix.com" [59]. In an attempt to ban "youtube.com", Pakistan telecom advertised invalid BGP routes which led a huge volume of traffic

not intended for Pakistan towards the Pakistan AS [159]. This a form of unintentional BGP hijacking.

There are many approaches in the literature to detect BGP anomalies such as time-series based, machine learning based and statistical pattern recognition based [5]. However, all of these approaches use statistical features derived from BGP updates within a specified time window as was similarly done in Section 5.1. Some examples are as follows:

- BGP message volume
- Number of announcement and withdrawals for an AS
- Number of new-path announcements
- Maximum and Average AS-Path length
- Number of duplicate withdrawals
- Packet size

Though the Bottom-Up approach is feasible, in the bigger picture as attacks evolve, a more robust approach or a feature set which remains invariant to attacks is necessary. In this section, invariant details of normal operations are captured and an anomaly detection model is trained using graphical information instead of statistical features.

5.2.2 Methodology

The core routers have connectivity information and they receive updates every few seconds announcing new paths or withdrawing existing ones. This connectivity information, and updates, represent a changing node and edge graph. Currently the core router only assimilates update information pertaining to its direct neighbours into its Global Routing Table (GRT). However the updates contain more information, and it is possible for the core router to build a view of the entire connected network as a

```
$ bgpctl show mrt file grt.updates.20-09-19_00_44
1600433040.328971 210.7.33.12[38022] -> 210.7.46.211[65002]: size 110 UPDATE
    Origin: 0
    AS-Path: 38022 3356 2914 11123
    Nexthop: 210.7.33.12
    Communities: 2914:410 2914:1005 2914:2000 2914:3000 3356:3 3356:22 3356:86 3356:575
    NLRI prefixes: 206.251.64.0/19 66.146.0.0/17
0.000004 2404:138:33:12::1[38022] -> 2404:138:46::211[65002]: size 44 UPDATE
    MP Unreach NLRI:  IPv6 unicast 2a0c:b641:26a::/48 2a09:4c2:36::/48
```

Figure 5.5: An example of a BGP update on a core router with the AS-Path highlighted.

network graph. Figure 5.5 shows an example of a BGP update on a core router. The *AS-Path* field are the AS numbers and NZ's core router's number is 38022. Each core router in the world has a unique AS number. The route from AS11123 would need to go through AS2914, AS3356 to reach AS38022. There is also an hourly view of the entire routing table and some of the lines are shown in Figure 5.6. The connections between ASs are highlighted. Using this information and the updates, a network graph can be built as show in Figure 5.7. Note that each core router will have a different view of the network. This depends on the arrival of the updates and the fact that each router is only concerned about its connections to the other ASs. This means to say that there may be connections between other ASs that the router is not aware off.

Upon building the entire network, the next step is to extract information or features which represent the graph. This is then used to train a ML model for anomaly detection. This approach has not been explored for BGP anomaly detection. BGP anomalies always manifest as changes in network graphs. During a BGP incident, large portions of network traffic will be rerouted leading to several nodes having more or less paths routing through them. Hence, the network graph will change substantially. Hence, this graphical approach remain invariant to different types of BGP anomalies. The graph so obtained is undirected, as edges (connections) link vertices (AS) symmetrically: meaning if two ASs are connected, the

```
TIME: 03/28/20 13:04:27
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 45.89.160.0/23
SEQUENCE: 8120
FROM: 210.7.33.5 AS38022
ORIGINATED: 03/15/20 07:13:26
ORIGIN: IGP
ASPATH: 38022 9500 4637 33891 205614
NEXT_HOP: 210.7.33.5
LOCAL_PREF: 100
COMMUNITY: 9500:900 38022:10800

TIME: 03/28/20 13:04:27
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 45.89.156.0/22
SEQUENCE: 8119
FROM: 210.7.33.12 AS38022
ORIGINATED: 03/05/20 20:04:23
ORIGIN: IGP
ASPATH: 38022 9500 4637 3257 28917 42946
NEXT_HOP: 210.7.33.12
LOCAL_PREF: 100
COMMUNITY: 9500:900 38022:10800

TIME: 03/28/20 13:04:28
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 110.195.32.0/21
SEQUENCE: 57318
FROM: 210.7.33.12 AS38022
ORIGINATED: 03/05/20 20:04:28
ORIGIN: IGP
ASPATH: 38022 9500 4637 58453 9808 9394 45069
NEXT_HOP: 210.7.33.12
LOCAL_PREF: 100
COMMUNITY: 9500:900 38022:10800

TIME: 03/28/20 13:04:30
TYPE: TABLE_DUMP_V2/IPV4_UNICAST
PREFIX: 204.75.67.0/24
SEQUENCE: 2530
FROM: 210.7.33.12 AS38022
ORIGINATED: 03/05/20 20:04:36
ORIGIN: IGP
ASPATH: 38022 9500 4637 7922 33491 26245 26245
NEXT_HOP: 210.7.33.12
LOCAL_PREF: 100
COMMUNITY: 9500:900 38022:10800
```

Figure 5.6: BGP table view on a core router generated every hour

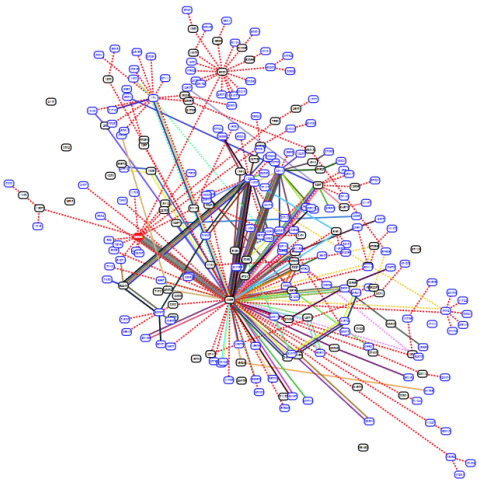


Figure 5.7: Network view of AS38022 at 0919 hours on 30 Aug 2020. The thickness of the lines represent multiple links and colours are for aesthetics.

traffic can flow in both directions.

Graph Representation

The next step is to represent the graph in a form for training a ML model. There are several ways to achieve this but not all of them are efficient or possible due to memory and computational limitations, as follows.

The most basic representation is the Adjacency matrix. The Adjacency matrix of a graph is a binary square matrix to indicate whether pairs of nodes are connected or not. Since the network graph is undirected, the Adjacency matrix is symmetric and has eigenvalues and eigenvectors. These information can be used as data representation of the graph to train the ML model. Unfortunately, computing the Adjacency matrix for the entire network has complexity $\mathcal{O}(n^2)$ where n is the number of vertices. This is not feasible in a live network with thousands of routers. Furthermore, since there are many vertices which are not connected, the Adjacency matrix is sparse and sparse matrices have increased time and memory complex-

ity [153].

Another representation is using *centrality* which describes how central each vertex (AS) is in the graph [33]. There are several centrality measures which can be used.

- **Degree centrality (DC)** of a vertex u is the degree of u (number of direct neighbours n) in the graph of $N-1$ other vertices.

$$DC(u) = \frac{n}{N-1}$$

- **Closeness centrality (CC)** of a vertex u is the reciprocal of the sum of the length of the shortest paths between u and all other vertices in the graph.

$$CC(u) = \frac{1}{\sum_{v \in V} d(u,v)}$$

where $d(u,v)$ is the shortest path in terms of number of links between vertices u and v. It is better to normalise this score to represent the average length of the shortest paths rather than the sum, giving

$$CC(u) = \frac{N-1}{\sum_{v \in V} d(u,v)}$$

where N is the total number of vertices in the graph.

- **Betweenness centrality (BC)** counts the fraction of shortest paths going through a vertex u.

$$BC(u) = \sum_{s \neq u \neq t} \frac{\gamma_{s,t}(u)}{\gamma_{s,t}}$$

where $\gamma_{s,t}(u)$ is the number of shortest paths between vertex s and vertex t that pass through u. $\gamma_{s,t}$ is the number of shortest paths between s and t.

- **Eigenvector centrality (EC)** generalizes degree centrality by incorporating the importance of the neighbors. The eigenvector centrality of vertex u is a function of its neighbors' centralities. It is proportional to the summation of their centralities.

$$EC(u) = \frac{1}{\delta} \sum_{j=1}^{n} A_{ij} EC(v_j)$$

where δ is a constant, n is the number of u's neighbours, v_j is the j^{th} neighbour of u and A is the Adjacency matrix.

Calculation of the Adjacency matrix and all possible paths between 2 vertices is computationally expensive. This rules out the use of BC and EC for the BGP application. DC is the easiest to compute and takes less than a second to process each update. CC represents the average inverse distance of a vertex to all its reachable neighbours and contains more information than DC. To compute the distance in CC, Dijkstra's algorithm is used [44] and takes around 6 minutes to process each update. This is still feasible and practical for BGP updates.

From Figure 5.7, few vertices are disconnected and this occurs occasionally at different time periods. A better measure of centrality of a vertex u is defined by equation 5.1 [199];

$$CC(u) = \frac{(n-1)}{(N-1)} \frac{(n-1)}{\sum_{v \in V} d(u,v)} \tag{5.1}$$

where $d(u,v)$ is the shortest path between vertices u and v. N is the total number of vertices in the graph and n is the total number of reachable vertices from u. A vertex u is reachable from another vertex v if there exists a path linking the two vertices [199].

Last but not least, Graph Neural Networks [210] can be used for representation learning of graphs. Each vertex aggregates feature information from its neighbours to compute a new feature vector layer by layer. After multiple iterations of information aggregation, the feature vector

of each vertex captures the structural information of its neighbourhood. This is used for vertex classification, link prediction, graph classification, etc. [197, 210]. Wang *et al.* [197] propose a one class classification framework for graph anomaly detection using Graph Neural Networks. This method accounts for feature information of each vertex. Though it can contain more information, it is not necessarily required for BGP anomaly detection at this stage. Multiple iterations also take longer time to process and may not be feasible in real-time. This method can be used if features such as the volume of traffic, wired or wireless connection at each core router is considered in building the anomaly detection model. It is hence, left as future work.

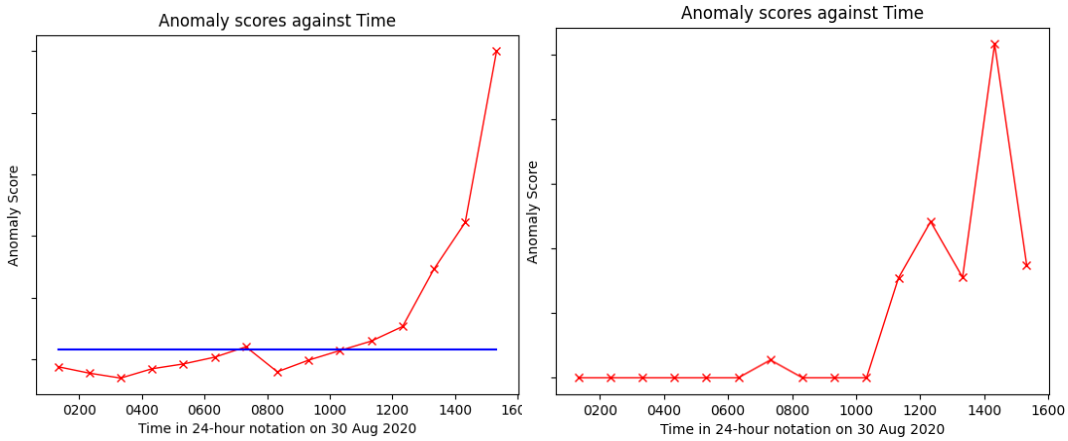
5.2.3 Experiments and Results

The incident of interest, called "Century Link Outage" occurred on 30 Aug 2020 [39]. Network operations centers around the world were alerted to it at 10:09 hours UTC through twitter. It was resolved around 1600 hours UTC. REANNZ [157] is New Zealand's National Research and Education Network. They monitor connections between research institutions within and out of New Zealand. They were alerted to this incident, again through twitter. Fortunately, during the period of downtime, New Zealand was not heavily affected as it was late in the night. The data provided by REANNZ on the New Zealand core router, AS38022, spans from 1400 hours 23 Aug 2020 to 1600 hours on 31 Aug 2020.

To further validate the technique and the detection of this incident, data from another core router in Japan called WIDE, AS2497, is obtained from *routeviews.org* during the same period [187]. The data spans from 0000 hours on 20 Aug 2020 to 0000 hours on 3 Sep 2020.

The centrality information calculated using (5.1) is used to train an AE as the offline model. Instead of training with all of the ASs in the network, only ASs in the first 2 hops of the respective core router, REANNZ and

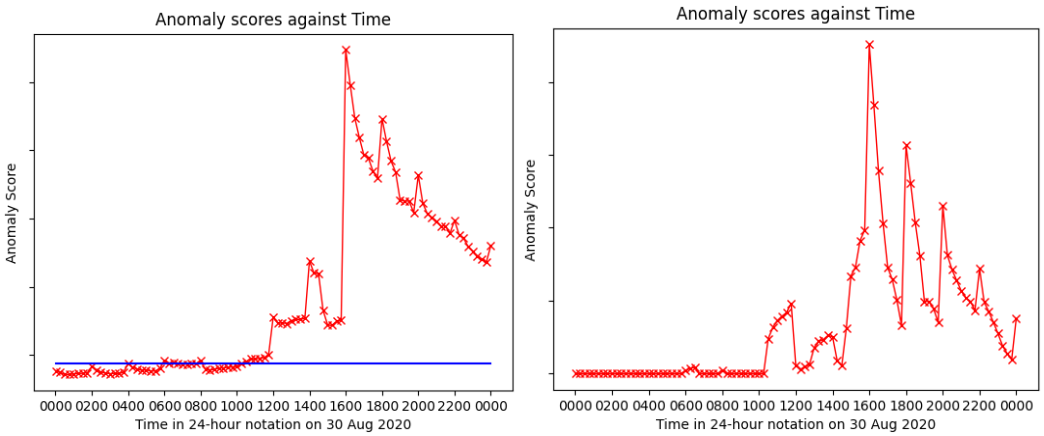
WIDE are taken. The details are similar to the model in Chapter 3. The Hybrid Online Offline framework is also compared with the purely offline AE. The online model in this instance is the IOCSVM. Data before 28 Aug is used to train the model, data on 29 Aug is used as a validation set. Data from 30 Aug is used as the test set. Figure 5.8 shows the performance of the offline AE and the Hybrid AE+MW+IOCSVM. Figure 5.9 shows the performance of the offline AE and the Hybrid AE+MW+IOCSVM on 30 Aug and for the next few days. The anomaly scores are on ordinal scale and thus, they are not comparable in between models. The threshold for the offline AE is determined based on the AE's performance on the validations set. For the AE+MW+IOCSVM model, anomaly scores greater than zero suggest that the point is above the threshold of the offline AE. A low anomaly score suggests that the point is an inlier and is potentially "new normal" data.



(a) Performance of AE (b) Performance of AE + MW + IOCSVM

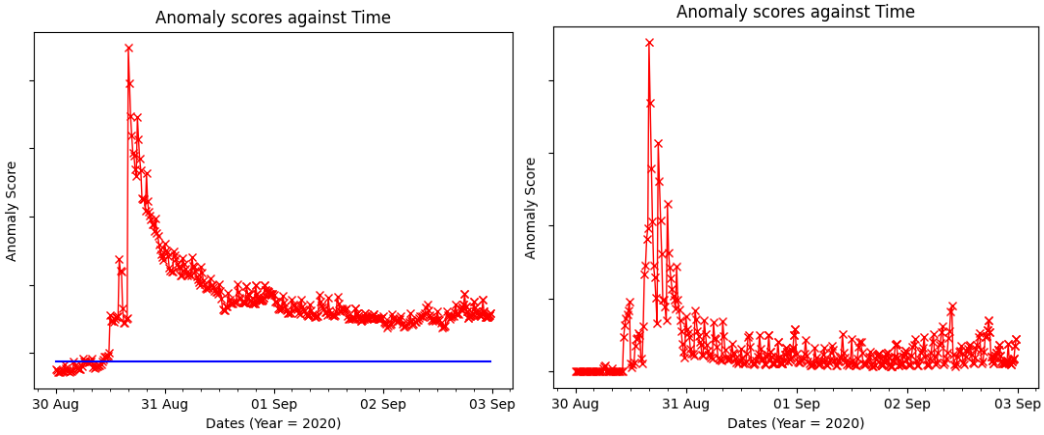
Figure 5.8: Anomaly scores of model at REANNZ core router of the BGP Century Link Outage. Threshold for offline AE determined based on validation set.

In both cases, the AE is enough to detect the anomaly as the score increases and breaches the threshold around the same time as the BGP



(a) Performance of AE on 30 Aug

(b) Performance of AE+MW+IOCSVM on 30 Aug



(c) Performance of AE from 30 Aug to 3 Sep

(d) Performance of AE+MW+IOCSVM from 30 Aug to 3 Sep

Figure 5.9: Anomaly scores of model at WIDE of the BGP Century Link Outage. Threshold for offline AE determined based on validation set.

updates arrive and change the centrality of the ASs. This can be seen in Figures 5.8a, 5.9a and 5.9c. However as the network resolves to normality, the anomaly score provided by the AE alone does not decrease because the network structure and the router's centralities have changed. The AE+MW+IOCSVM model in Figures 5.8b, 5.9b and 5.9d show that the new structures have lower anomaly scores as they are now "inliers" with respect to the online model. After the Century link outage, as the network resolves itself, there are also new ASs within the first 2 hops of the WIDE router. This indicates that the network graph has changed and as such the anomaly score is not zero. However, they are low and stable to indicate that this new network centralities are inliers. The variation of the anomaly scores after the anomalous event suggest that there is new information which needs to be incorporated into the offline model. From Figure 5.10, the MW-test are rejected yet, the median of the anomaly scores are stable and the anomaly scores are low from Figure 5.9d. As described in Chapter 3, clustering can be used to determine the new centrality values to update the offline model. From Figures 5.11 and 5.12, both REANNZ's and WIDE's core routers' view of the network has slightly changed.

Hence, in this section, a novel method of using centralities of network connections as features to detect BGP anomalies was explored. It was also shown that new normal is an important scenario to consider. The Hybrid Online Offline Framework is also validated.

The next section shows how to extract useful information from video surveillance stream to build a simple yet effective anomaly detection model.

5.3 Surveillance monitoring

Monitoring surveillance streams is a bothersome job which should be automated. There are many different surveillances looking out for different things such as road, airport, maritime, store, coastal, forest, ocean, etc. [63]. The main challenge in this domain is to be able to extract useful informa-

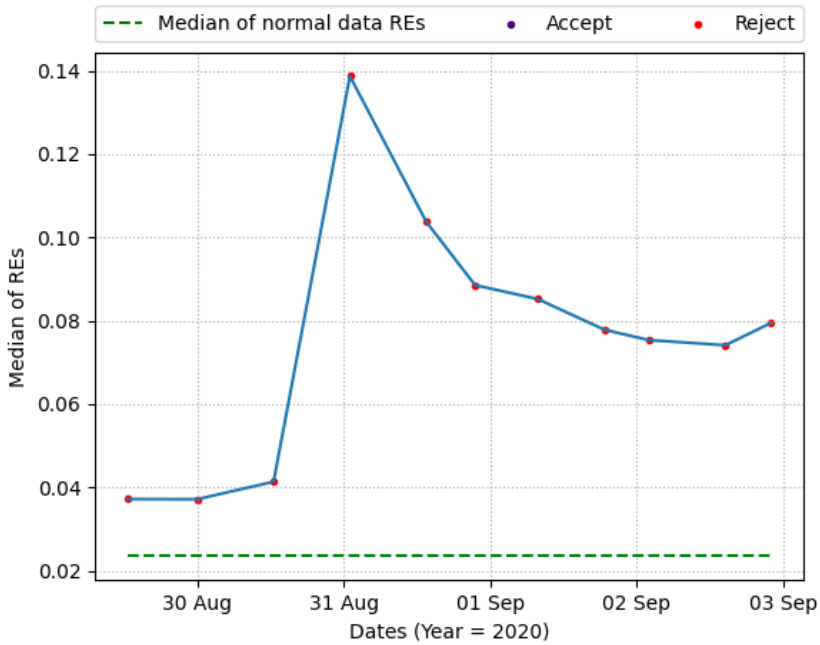
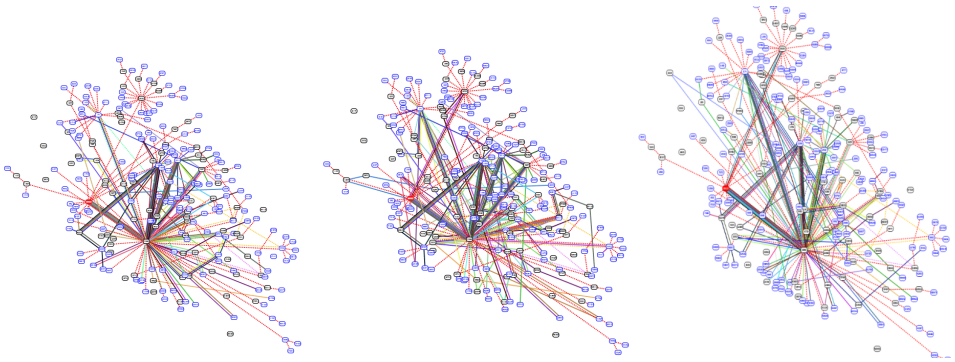
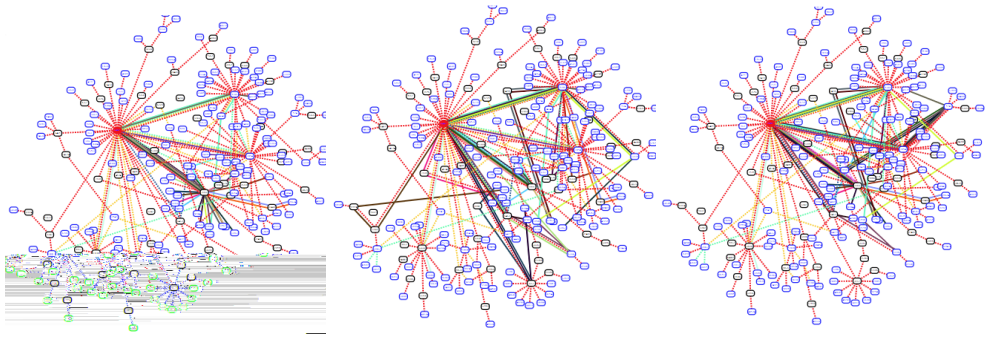


Figure 5.10: MW tests of the AE+MW+IOCSVM on WIDE's core router data.



(a) 0919 hours (b) 1019 hours (c) 1619 hours

Figure 5.11: Network view of REANNZ core router, AS38022 at (a) 0919 hours before the outage, (b) 1019 hours during the outage and (c) 1619 hours after the outage on 30 Aug 2020. The thickness of the lines represent multiple links and colours are for aesthetics.



(a) 0919 hours (b) 1019 hours (c) 1619 hours

Figure 5.12: Network view of WIDE core router, AS2497 at (a) 0919 hours before the outage, (b) 1019 hours during the outage and (c) 1619 hours after the outage on 30 Aug 2020. The thickness of the lines represent multiple links and colours are for aesthetics.

tion from the video stream to build an anomaly detection model. The following section investigates this with the scenario of accident detection in road traffic surveillance streams.

5.3.1 Introduction

Detecting accidents quickly can enable emergency services to be deployed in time. There are a few studies which have tackled this problem [13, 45, 148]. The method by Doshi *et al.* [45] is fast but it does not capture the motion of the object. Peri *et al.* [148] use post processing to determine anomalies and not in real-time. Bai *et al.* [13] analyzes each location in the video instead of vehicle and its trajectory to detect anomalies. All of the above work involves a pipeline to extract information. Similarly, this section presents a novel pipeline, with different elements to obtain details such as motion information in the video stream before training an AE as an anomaly detector. The EP-BRVFL-AE model developed in Chapter 4 is validated as the anomaly detector as it is simple and has low computational and memory complexities.

5.3.2 Methodology

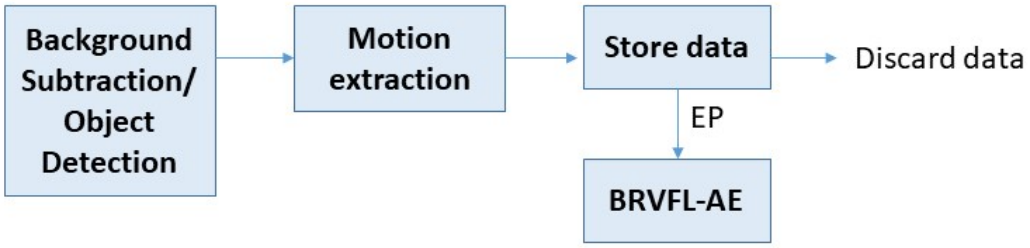


Figure 5.13: Pipeline for extracting information from surveillance video streams.

Figure 5.13 shows the required pipeline to extract information from a surveillance stream to train an anomaly detection model. This work explores off the shelf methods for each aspect of the pipeline to validate the ideas introduced in the earlier chapters.

Background Subtraction and Object Detection

One commonly used method is an adaptive GMM [87]. This method places a GMM to model each pixel in the video frame over a few frames. Each mixture is ordered and from the first few strongest mixtures, pixels that are above 2.5 standard deviations away from the selected mixtures are marked as a foreground. The GMM is updated with each frame. This method is effective but in real-time stationary objects will enter the background during the update. An alternative is to train a GMM similarly with enough video frames to obtain the best model and take the mixture with the highest weight as the background [69]. With enough frames, stationary objects will not be in the background. The training however, needs to be performed offline but upon which, in deployment, no update is required. Recently, VAE has also been suggested for background modelling [55]. However, this requires too much computational power for near similar

gains as the GMM. All of them have been explored to obtain near similar results.

Upon using a Background Subtraction (BS) method, all foreground objects can be identified. However, using only BS is noisy and when objects move in front of one another, the method identifies 2 objects as one. To address this, existing Object Detection (OD) methods such as YOLO(v3) [53], Faster R-CNN [156], SSD [108], amongst others can be used. One other caveat is that these models require supervised pre-training and thus, are only able to detect objects in the training dataset. For the most part, this is not a major issue in traffic surveillance as objects are usually vehicles. Due to the deep nature of the neural network involved in these models however, inference takes longer ($\approx 1.5s$). In this pipeline YOLO(v3) has been utilised because its average precision is on par with the rest and it is faster [53]. However, the choice of BS method or object detection model is not the focus of this thesis. The main goal is to obtain the bounding boxes for each of the objects detected in the video stream.

Motion Extraction

Learning the motion of the objects in the stream is important so as to detect anomalous motions. It is also important to know where in the frame the specific motion occurs. Motion information includes the location, size, speed and direction of the object. Two widely used methods are based on optical flow [54, 110]. The Lucas-Kanade method finds a few points such as corners of the object and calculates the horizontal and vertical change in pixel intensity between frames [110]. Capturing the corners of the object poses another challenge with this method. The Farneback optical flow computes this change over all points in the frame [54]. These vectors are then binned into a histogram where the bins are direction of motion. With multiple objects in the frame moving in different directions, this method will not work very well.

Object tracking methods can also be used to extract motion. The Simple

Online and Realtime Tracking (SORT) method makes use of the Hungarian algorithm and Kalman filter to track and assign velocities given the bounding box of the object [16]. The advanced version of SORT called, Deep SORT uses deep learning to integrate the appearance information to improve performance [202]. This requires pre-training with the type of objects that are to be tracked.

In this section the SORT method is used. The bounding boxes of the detected object are fed into the SORT algorithm to obtain 7 values for each object in the frame as shown in 5.2.

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}] \tag{5.2}$$

These 7 values represent the state of the object in the frame. u and v represent the horizontal and vertical pixel location of the centre of the object. s and r represent the scale (area) and aspect ratio of the object's bounding box. The aspect ratio is considered to be constant. When a detection is associated to a target, the detected bounding box is used to update the target state where the velocity components are solved optimally via a Kalman filter framework [88]. If no detection is associated to the target, its state is simply predicted without correction using the linear velocity model.

Using EP for Online Learning

One of the motivations as highlighted in Chapter 2 for online learning is to capture concept drift. Chapter 3 highlighted that new information not seen during training can occur and to build a robust model, one needs to incorporate this into the model. Online learning is similar to addressing catastrophic forgetting [96], i.e. to update a model without having to completely retrain it. There have been many studies in this area [91, 188]. This is a whole area of research and it is beyond the scope of this thesis. Kirkpatrick *et al.* [96] suggest Elastic Weights Consolidation where the new weights lie in the intersection of the existing posterior distribution of a model and the posterior distribution of the new task or information. This

Bayesian approach is similar to what was achieved in Section 4.2.5 and Figure 4.7b. In Section 4.2.5, the experiment evaluated the asynchronous scenario where the updates arrived one by one. In essence, this is similar to online learning where new information is obtained later. In relation to Kirkpatrick *et al.* [96], if such an intersection between the posterior distribution do not exist, then it justifies having to increase the number of parameters, i.e. using a wider or deeper neural network.

This is especially relevant for Edge AI where the edge device such as the camera does not have enough memory to store a lot of data. As shown in Figure 5.13 and Figure 4.10, data is discarded after using it for training the BRVFL-AE with EP being used to incorporate new information.

5.3.3 Experiments and Results

Data from the 2021 AI city challenge on detecting anomalies such as crashes, stalled vehicles is used for this experiment [129]. The data is collected from a video surveillance camera at a highway. Two frames from the video are shown in Figure 5.14.

(a) Normal traffic (b) Crash

Figure 5.14: Frames during normal traffic and a crash

Firstly YOLO(v3) is used to find the bounding boxes of the vehicles. YOLO(v3) was pretrained on the COCO dataset [106]. Using the SORT algorithm, 7 values as shown in Equation 5.2 are obtained for each object

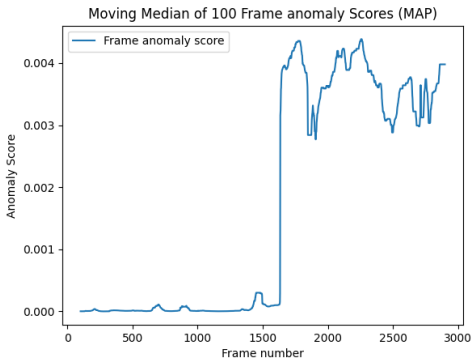
and these are taken as the training data for BRVFL-AE. The trained model is evaluated thereafter with normal traffic and when the crash occurs.

There are multiple objects in each frame with each giving its own anomaly score. Since the number of objects do not remain the same, taking average dilutes the score of frame suppose it has an anomalous object. Hence, the maximum anomaly score of all objects in the frame is taken as the frame anomaly score. One other issue was that "trucks" on the highway gave high frame anomaly scores because they varied in sizes and shapes and there was not enough "truck" information in the training set. In real-world implementation, this is a likely scenario. Fortunately, "trucks" are not present in all frames; and when an accident occurs, all frames from that period should have higher anomaly scores. Considering "trucks" as noise, the frame anomaly scores are smoothed using a moving median filter over 100 frames. The results are shown in Figure 5.15.

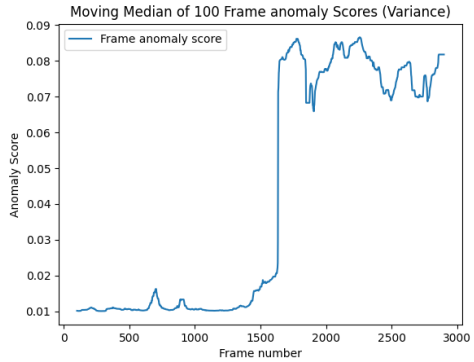
In the video, the accident occurs on the 1400^{th} frame. With the moving median filter, the accident is detected on the 1635^{th}, 1614^{th} and 1634^{th} frame where the scores jump up. A threshold can be easily drawn to detect this. This is about 8 seconds later for a video stream at 30 frames per second. YOLO also takes a while to perform inference per frame. To improve the speed, all frames in the video need not be considered because the information between subsequent frames are very similar. YOLO(v3) though reported as the fastest object detection method [53], it takes more than 1 second to detect all of the objects in the frame. More research in this area is required and it is described in Section 6.2.

Furthermore, if there are a lot of noise and smoothing methods need to be used, it will reduce the real world feasibility of the method. Thus, training the BRVFL-AE with more data is also necessary. However, since the pipeline is entirely automated, if the object detection method captures "objects" which are not relevant, this will affect the BRVFL-AE. Thus, the accuracy of the object detection model matters. Improving this accuracy can be explored through combining background subtraction and object

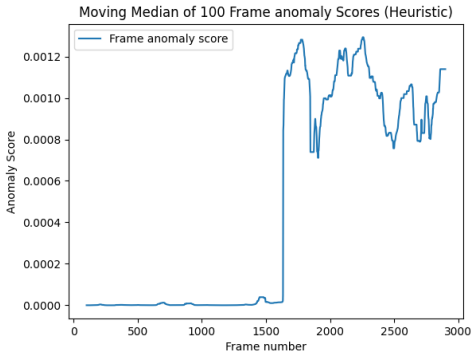
detection [60].



(a) MAP

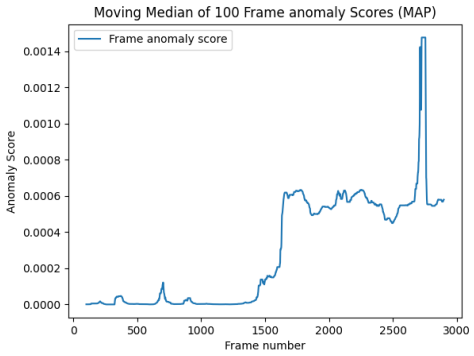


(b) Variance

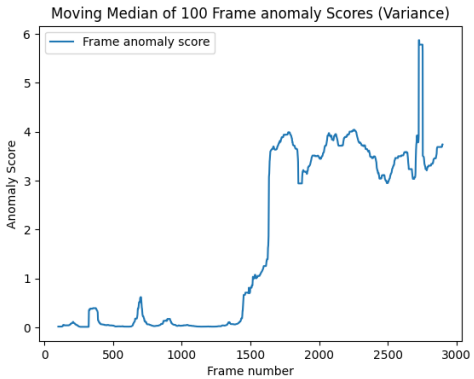


(c) Heuristic

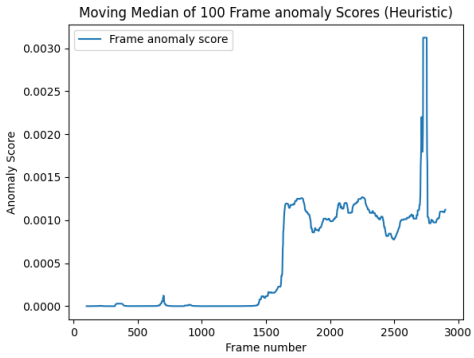
Figure 5.15: Frame anomaly score from the BRVFL-AE model with (a) MAP estimate, (b) Variance and (c) Heuristic measure.



(a) MAP



(b) Variance



(c) Heuristic

Figure 5.16: Frame anomaly score from the EP-BRVFL-AE trained with 4 separate partitions. (a) MAP estimate, (b) Variance and (c) Heuristic measure.

Since edge devices are typically memory-limited, the EP-BRVFL-AE was implemented for online learning in this application. As was shown in Section 4.2.5 and Figure 4.7b, EP can be used to update the BRVFL-AE. The training dataset is split into 4 equal components and combined one by one using EP. The results are shown in Figure 5.16. Though scores vary, as the model is different from the one trained with all the data, the effectiveness in detecting the crash is retained. This shows that it is possible for the model to be updated using EP. However, more research is required to ascertain whether this type of one-shot learning is sufficient in the long run.

In this section a novel pipeline is developed to be able to extract information from video streams for ML and anomaly detection thereafter. The EP-BRVFL-AE was again validated as a simple model which can nonetheless perform effectively in a real world scenario.

5.4 Summary

The challenges of three unique real world domains were addressed with three scenarios: namely systems monitoring applied to an ISP, network graphs monitoring applied to BGP updates and surveillance monitoring applied to road traffic video stream. For systems monitoring, the main challenge was to identify the right features and the Bottom-Up approach was devised. To detect anomalous BGP updates, network graph information was parsed using centrality values of nodes to build an anomaly detection model. Lastly, a novel pipeline was devised to extract information to build a simple yet effective accident detection model. The methods devised in this chapter are widely applicable to several applications within each domain.

The next chapter summarises the entire thesis by highlighting the contributions and future work.

Chapter 6

Conclusion

The goal of this thesis was to develop methods for anomaly detection in dynamic and distributed scenarios. This chapter summaries the thesis, outlines the major contributions, and points out the directions for future work.

6.1 Contributions

The main research question "*How can anomaly detection take into consideration evolving and heterogeneously distributed data?*" was answered in two parts, in Chapters 3 and 4. Then Chapter 5 built novel methods to address the anomaly detection problem in each domain and implemented the methods developed in the earlier chapters. Specifically:

1. **A Hybrid Online Offline framework was designed to achieve robust anomaly detection system under evolving data.**
   The offline model learns existing normal data well and the online model performs outlier detection. The hybrid framework exploits the strengths of the individual models and was shown to outperform either individual model. This was done using the offline model's learned knowledge as a bias for the online model to select data to

train on in the data stream. A dynamic median threshold and the MW-test were employed to prevent the online model from learning on anomalous data. The framework was evaluated on public streaming datasets: the UNSW-NB15, CTU13-13, CTU13-9, CTU13-1 and CTU13-3. A combination of three offline models and six online models were considered in the framework [142, 144].

From this behaviour-based version, it was shown that clustering can be used to determine whether there are new types of normal data. This is only possible based on the anomaly scoring of the designed Hybrid Online Offline model. This is because the hybrid framework outputs anomaly scores in the same range for new unknown data points. Upon detection of new normal data, the offline model can be retrained or updated using EP if the offline model is Bayesian.

A signature-based method using an offline Rad-NN and an online SVM has also been designed. This also outperforms individual offline or online models. The method uses confidence scoring to determine the confidence of each of the individual model's label of the incoming data. To measure the confidence of the Rad-NN, a novel confidence measure was formulated in Equation (3.1) [141, 143].

2. **EP was used to train a BRVFL-AE with heterogeneously distributed data. The method was further explored with a deep BAE trained using VI.**

Amongst the methods available for distributed training, EP and BRVFL-AE were chosen because of the following benefits. Firstly, BRVFL-AE is a SLFN which provides a closed form solution. Secondly, it is quick to train. Thirdly, the Bayesian method provides a way to measure uncertainty. Fourthly, EP allows the aggregating of parameter val-

ues from different sites while accounting for covariance. With these benefits, the method can be implemented on edge devices such as cameras or routers where computation capabilities are limited. The method was evaluated on six machine learning datasets: UNSW-NB15, NSLKDD, Shuttle, Abalone, Australian Credit Approval and Pageblocks. The method was further tested under a scenario where the data were segregated in an unevenly and biased manner. This was achieved by separating the data via a GMM and each site contained data from one mixture. The method was also evaluated under different network densities and asynchronous updates. The method was shown to perform well under all of the above mentioned scenarios.

It was also shown that EP can work with a deep BAE to build the offline model in a distributed manner, as part of the Hybrid Online-Offline Framework. For this to be feasible, there are caveats such as the use of VI and the need for greater computational resources. As the method was also shown to work with asynchronous updates, it can be adapted to perform online learning with new data.

3. **Three real world domains were explored using three scenarios respectively.**

   - *Systems monitoring applied to an ISP*. The main challenge was to determine the appropriate variables to measure to build an anomaly detection model. A novel Bottom-Up approach was introduced and subsequently, an iterative procedure was designed to measure the right variables. Feature identification methods were used to characterise anomalous behaviours instead of using it before training a ML model as done in the literature. The Bottom-Up approach is a generic method which

can be used to identify features when there are an abundant of variables to monitor.

The Hybrid framework was also validated and has shown to perform better than a single model. The model was evaluated using real-world systems monitoring data from an ISP.

- *Network Graphs monitoring applied to BGP updates.* A novel method of formulating network graphs from the BGP updates and representing the graph using centrality information was proposed. This method captures the invariants of the domain. The centrality information of each node is then used to represent the graph and train an anomaly detection model. Data from RE-ANNZ and the WIDE core router were used. This method of expressing connections as graphs and using centrality values can also be used for building anomaly detection model in other networked scenarios such as power grids.

  The Hybrid framework was also validated and has shown to perform better than a single model. There are also many ways the routers can be connected in normal scenarios. The Hybrid framework also identifies new normal scenarios with the MW-tests which can be used to update the offline model.

- *Surveillance monitoring applied to road traffic video stream.* A novel pipeline is designed to extract pertinent information required to detect anomalies and accidents. Firstly, the background subtraction is employed using GMM [69]. Next, YOLO(v3) [53] is used to detect objects to fine tune the bounding boxes. Using SORT: a combination of Kalman Filter and the Hungarian algorithm [16], the motion and size of the object are extracted from subsequent frames. This information is used to train an EP-BRVFL-AE. Data from the 2021 AI city challenge were used [129]. It detects an accident on the highway successfully. This generic pipeline can be used in other application scenarios such

as pedestrian footpath or security monitoring. EP was further tested to perform online learning with batches of data and similar results were achieved.

6.2 Future Work

Though the contributions and experiments have addressed the objectives, they have opened up many avenues for future work, to expand the research in both academic and practical scenarios.

6.2.1 Hybrid Model

The Hybrid Online Offline framework is very flexible. Firstly, using smaller values of θ for the MW test helps to identify batches with more anomalous data. Secondly, the re-training criteria can be a function of other attributes in the network instead of reaching a stipulated size or seasonality. Thirdly, with seasonality, the percentage of expected normal data need not be fixed. The offline model can be trained with other loss functions (see [27]) for different representations of the data. Hence, the framework can be adapted for different scenarios in the real world.

This thesis introduced a heuristic search for new data using clustering in Section 3.3 and Algorithm 2. This method can be extended to search for *inliers*. Inliers can either refer to data which lie in the interior of a statistical distribution, or to data generated by different mechanisms where one is hidden in an area of high density of the other [52]. Since they occur together after removing known normal data, these can be classified as new normal or anomalies which have a particular distribution such as DoS attacks. If the inliers are new information, they can be incorporated into the offline model. The offline model can be retrained completely, or if it is a BNN, EP can be used. It has been shown in Figure 4.7 that EP can be used to build a model with batches of data arriving one after the other.

However, distinguishing inliers as new normal or anomalies is not obvious, which leads to a future research direction. With a human observer, the Bottom-Up approach and iterative procedure can be used to craft new variables such that the occurrence can be separated in future.

Another direction is being able to incorporate new variables into an already trained model. Chen *et al.* [35] increased the size of the hidden layers of a neural network, based on the concept of function preserving transformation. However, the authors do not explore a new input variable. Developing a method to incorporate a new variable into an already trained NN would help speed up the iterative procedure from Section 5.1 instead of having to retrain the offline model from scratch. The applications for this area of research are also vast. For example, after adding sensors in industrial monitoring or in autonomous vehicles to measure new variables, the already trained model can be updated.

In the Hybrid model, the MW-test is used to test whether the mean ranks of distribution of the anomaly scores have significantly changed. Other methods such as the Kolmogorov–Smirnov (KS) test [116], relative entropy (Kullback-Leibler Divergence) [100], Welch's T-test [200] have been explored but were not as effective as the MW-test for the Hybrid framework. The KS test was too sensitive to small changes, relative entropy required determining appropriate intervals while the Welch's T test required the Gaussian assumption. One avenue to explore is to determine how exactly the distribution of the data has changed in the online scenario, given the MW-test result. Similarly in the Signature-based hybrid model in Section 3.4, the distribution of the confidence scores can be tested to show when the offline Rad-NN needs to be updated.

6.2.2 Bayesian Method for Distributed Learning

The method of training the EP-BRVFL-AE can be further extended with more parameters. For each dimension j and each weight, a value of σ_j^2

and γ_l can be explored instead of one value σ^2 and γ which were determined based on a hyperprior. The former will lead to a different covariance matrix Σ_j for each dimension and the latter is also known as Automatic Relevance Determination (ARD) [20,164]. Also, instead of assuming a Gaussian distribution for the likelihood and prior in (4.6) and (4.7), other distributions such as the Student's t-distribution can be used. However these come with more computational requirements.

It was shown in the experiments of Section 5.3 that EP can be used as a tool for online learning. More research is required to ascertain whether this type of One-shot learning [57] is sufficient and whether it can work with deeper NNs.

6.2.3 Real World Scenarios

To implement the methods, hardware compatibility must be determined. The number of CPU or GPU cores and RAM required for training the model or storing data or model parameters must fit within the performance requirements of speed and accuracy. This study is immensely important in all of the real world scenarios.

In the systems monitoring situation, as the antenna site was in a rural area, power consumption of running the algorithm must also be considered. Implementing a solution to address one problem might lead to new problems.

For network graphs monitoring, the use of graphical neural networks can be explored. Other forms of centrality measures such as Approximate Betweenness Centrality can also be considered in representing the network graph for training the anomaly detection model [15, 149]. This may be quicker to process each BGP update. Another possible direction would be to train a model at each AS based only on its own centrality values. Then when anomalous updates occur, the anomalous AS(s) can be detected quickly and tracked. This may prevent anomalous updates from

disrupting the entire network.

For surveillance monitoring implementation, YOLO(v3) takes too long to detect all of the objects. The depth of the YOLO neural network can be reduced if it is trained to only detect a few relevant classes of objects instead of all 91 classes [106]. Alternatively, transfer learning methods can be used to train a smaller neural network which provides similar results [12, 35, 76]. This would save memory in the edge device and time during forward propagation at inference.

6.3 Final Conclusion

With the continual advancement in technology and applications, anomaly detection will always be an important research area. For example, detecting anomalies in a distributed sense became more important as IoT and Big Data took off. This thesis has worked with two main theoretical scenarios which are the root of many of the practical problems found in the real world. These are dynamic data streams and distributed scenarios.

This research has attempted to capture the core problem of extracting new information from dynamic data streams, and has used that to improve anomaly detection performance. The Hybrid framework can still be improved as shown in Section 6.2. For distributed scenarios, a Bayesian approach was implemented and this thesis has shown that this direction can yield many potential gains.

There is no one best solution for the problem of anomaly detection, as was illuminated by the case studies explored here. Even though at its core, the question is always the same, the answer demands engagement with the fundamentals and context that are unique to each domain.

Bibliography

[1] ACEVEDO-VALLE, J. M., TREJO, K., AND ANGULO, C. Multivariate Regression with Incremental Learning of Gaussian Mixture Models. In *CCIA* (2017), pp. 196–205.

[2] ADUROJA, A., SCHIZAS, I. D., AND MAROULAS, V. Distributed principal components analysis in sensor networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), IEEE, pp. 5850–5854.

[3] AGGARWAL, C. C. Outlier analysis. In *Data mining* (2015), Springer, pp. 237–263.

[4] AGGARWAL, C. C., AND YU, P. S. Outlier detection for high dimensional data. In *ACM Sigmod Record* (2001), vol. 30, ACM, pp. 37–46.

[5] AL-MUSAWI, B., BRANCH, P., AND ARMITAGE, G. BGP anomaly detection techniques: A survey. *IEEE Communications Surveys & Tutorials 19*, 1 (2016), 377–396.

[6] ALAEI, P., AND NOORBEHBAHANI, F. Incremental anomaly-based intrusion detection system using limited labeled data. In *Web Research (ICWR), 2017 3th Int'l. Conf. on* (2017), IEEE, pp. 178–184.

[7] ALMALAWI, A., FAHAD, A., TARI, Z., ALAMRI, A., ALGHAMDI, R., AND ZOMAYA, A. Y. An efficient data-driven clustering tech-

nique to detect attacks in SCADA systems. *IEEE Trans. on Information Forensics and Security 11*, 5 (2015), 893–906.

[8] ALRAWASHDEH, K., AND PURDY, C. Fast activation function approach for deep learning based online anomaly intrusion detection. In *4th Int'l. Conf. on Big Data Security on Cloud, Int'l. Conf. on High Performance and Smart Computing, and Int'l. Conf. on Intelligent Data and Security* (2018), IEEE, pp. 5–13.

[9] ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician 46*, 3 (1992), 175–185.

[10] ATAKLI, I. M., HU, H., CHEN, Y., KU, W. S., AND SU, Z. Malicious node detection in wireless sensor networks using weighted trust evaluation. In *Proceedings of the 2008 Spring simulation multiconference* (2008), pp. 836–843.

[11] AYGUN, R. C., AND YAVUZ, A. G. Network anomaly detection with stochastically improved autoencoder based models. In *4th Int'l. Conf. on Cyber Security and Cloud Computing* (2017), IEEE, pp. 193–198.

[12] BA, J., AND CARUANA, R. Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems* (2014), Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc.

[13] BAI, S., HE, Z., LEI, Y., WU, W., ZHU, C., SUN, M., AND YAN, J. Traffic Anomaly Detection via Perspective Map based on Spatial-temporal Information Matrix. In *CVPR Workshops* (2019), pp. 117–124.

[14] BELLMAN, R. E. *Adaptive control processes*. Princeton university press, 2015.

[15] BERGAMINI, E., MEYERHENKE, H., AND STAUDT, C. L. Approximating betweenness centrality in large evolving networks. In *2015 Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments (ALENEX)* (2014), SIAM, pp. 133–146.

[16] BEWLEY, A., GE, Z., OTT, L., RAMOS, F., AND UPCROFT, B. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)* (2016), IEEE, pp. 3464–3468.

[17] BHUYAN, M. H., BHATTACHARYYA, D. K., AND KALITA, J. K. Survey on incremental approaches for network anomaly detection. *arXiv preprint arXiv:1211.4493* (2012).

[18] BIGDELI, E., MOHAMMADI, M., RAAHEMI, B., AND MATWIN, S. Incremental anomaly detection using two-layer cluster-based structure. *Information Sciences 429* (2018), 315–331.

[19] BISHOP, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural computation 7*, 1 (1995), 108–116.

[20] BISHOP, C. M. Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA* (2006).

[21] BLEI, D. M., JORDAN, M. I., ET AL. Variational inference for Dirichlet process mixtures. *Bayesian analysis 1*, 1 (2006), 121–143.

[22] BLEI, D. M., KUCUKELBIR, A., AND MCAULIFFE, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association 112*, 518 (2017), 859–877.

[23] BLUNDELL, C., CORNEBISE, J., KAVUKCUOGLU, K., AND WIERSTRA, D. Weight uncertainty in neural network. In *International Conference on Machine Learning* (2015), PMLR, pp. 1613–1622.

[24] BOUTABA, R., SALAHUDDIN, M. A., LIMAM, N., AYOUBI, S., SHAHRIAR, N., ESTRADA-SOLANO, F., AND CAICEDO, O. M. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Jrnl. of Internet Services and Applications 9*, 1 (2018), 16.

[25] BRAHMA, P. P., WU, D., AND SHE, Y. Why deep learning works: A manifold disentanglement perspective. *IEEE Transactions on Neural Networks and Learning Systems 27*, 10 (2015), 1997–2008.

[26] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. LOF: identifying density-based local outliers. In *sigmod record* (2000), vol. 29, ACM, pp. 93–104.

[27] CAO, V. L., NICOLAU, M., AND MCDERMOTT, J. Learning Neural Representations for Network Anomaly Detection. *IEEE Trans. on Cybernetics 49*, 8 (Aug 2019), 3074–3087.

[28] CARLINI, N., AND WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017), pp. 3–14.

[29] CHAHAL, K. S., GROVER, M. S., DEY, K., AND SHAH, R. R. A hitchhiker's guide on distributed training of deep neural networks. *Journal of Parallel and Distributed Computing 137* (2020), 65–76.

[30] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[31] CHANDRASHEKAR, G., AND SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering 40*, 1 (2014), 16–28.

[32] CHAPELLE, O. Training a support vector machine in the primal. *Neural computation 19*, 5 (2007), 1155–1178.

[33] CHEN, H., YIN, H., CHEN, T., NGUYEN, Q. V. H., PENG, W.-C., AND LI, X. Exploiting centrality information with graph convolutions for network representation learning. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), IEEE, pp. 590–601.

[34] CHEN, T., FOX, E., AND GUESTRIN, C. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning* (2014), pp. 1683–1691.

[35] CHEN, T., GOODFELLOW, I., AND SHLENS, J. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641* (2015).

[36] CHENAGHLOU, M., MOSHTAGHI, M., LECKIE, C., AND SALEHI, M. Online clustering for evolving data streams with online anomaly detection. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining* (2018), Springer, pp. 508–521.

[37] CHIB, S., AND GREENBERG, E. Understanding the metropolis-hastings algorithm. *The american statistician 49*, 4 (1995), 327–335.

[38] CLARK, D. R., AND THAYER, C. A. A primer on the exponential family of distributions. In *Casualty Actuarial Society Spring Forum* (2004), pp. 117–148.

[39] CNN. Major internet outage: Dozens of websites and apps were down. *https://edition.cnn.com/2020/08/30/tech/internet-outage-cloudflare/index.html* (2020).

[40] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[41] DAVIS, J., AND GOADRICH, M. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (2006), pp. 233–240.

[42] DE CASTRO, L. N., CASTRO, L. N., AND TIMMIS, J. *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media, 2002.

[43] DHANABAL, L., AND SHANTHARAJAH, S. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int'l. Jrnl. of Advanced Research in Computer and Communication Engineering 4*, 6 (2015), 446–452.

[44] DIJKSTRA, E. W., ET AL. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[45] DOSHI, K., AND YILMAZ, Y. Fast unsupervised anomaly detection in traffic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), pp. 624–625.

[46] DROMARD, J., ROUDIÈRE, G., AND OWEZARSKI, P. Online and scalable unsupervised network anomaly detection method. *IEEE Trans. on Network and Service Management 14*, 1 (2017), 34–47.

[47] DUA, D., AND GRAFF, C. UCI Machine Learning Repository, 2017.

[48] DUAN, M., LIU, D., CHEN, X., LIU, R., TAN, Y., AND LIANG, L. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems 32*, 1 (2020), 59–71.

[49] DUCH, W. Filter methods. In *Feature Extraction*. Springer, 2006, pp. 89–117.

[50] EL-SAYED, H., SANKAR, S., PRASAD, M., PUTHAL, D., GUPTA, A., MOHANTY, M., AND LIN, C.-T. Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing environment. *IEEE Access 6* (2017), 1706–1717.

[51] ERFANI, S. M., RAJASEGARAR, S., KARUNASEKERA, S., AND LECKIE, C. High-dimensional and large-scale anomaly detection using a linear OCSVM with deep learning. *Pattern Recognition 58* (2016), 121–134.

[52] FALKENHAGEN, U., KÖSSLER, W., AND LENZ, H.-J. A Likelihood Ratio Test for Inlier Detection. In *Workshop on Stochastic Models, Statistics and their Application* (2019), Springer, pp. 351–359.

[53] FARHADI, A., AND REDMON, J. Yolov3: An incremental improvement. *Computer Vision and Pattern Recognition* (2018).

[54] FARNEBÄCK, G. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis* (2003), Springer, pp. 363–370.

[55] FARNOOSH, A., REZAEI, B., AND OSTADABBAS, S. DeepPBM: deep probabilistic background model estimation from video sequences. *arXiv preprint arXiv:1902.00820* (2019).

[56] FAY, M. P., AND PROSCHAN, M. A. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys 4* (2010), 1.

[57] FEI-FEI, L., FERGUS, R., AND PERONA, P. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence 28*, 4 (2006), 594–611.

[58] FISCHER, L., HAMMER, B., AND WERSING, H. Combining offline and online classifiers for life-long learning. In *IJCNN* (2015), pp. 1–8.

[59] FOR NAMES, I. C. A., AND NUMBERS, I. Domain Name Hijacking A Preliminary Report. *https://www.icann.org/en/system/files/files/domain-hijcaking-mdp-05apr05-en.pdf* (2005).

[60] FU, Z., CHEN, Y., YONG, H., JIANG, R., ZHANG, L., AND HUA, X.-S. Foreground gating and background refining network for surveillance object detection. *IEEE Transactions on Image Processing 28*, 12 (2019), 6077–6090.

[61] GAL, Y., AND GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (2016), PMLR, pp. 1050–1059.

[62] GARCIA, S., GRILL, M., STIBOREK, J., AND ZUNINO, A. An empirical comparison of botnet detection methods. *computers & security 45* (2014), 100–123.

[63] GARCIA-GARCIA, B., BOUWMANS, T., AND SILVA, A. J. R. Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review 35* (2020), 100204.

[64] GELMAN, A., CARLIN, J. B., STERN, H. S., DUNSON, D. B., VEHTARI, A., AND RUBIN, D. B. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[65] GELMAN, A., VEHTARI, A., JYLÄNKI, P., SIVULA, T., TRAN, D., SAHAI, S., BLOMSTEDT, P., CUNNINGHAM, J. P., SCHIMINOVICH, D., AND ROBERT, C. Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. *arXiv preprint arXiv:1412.4869* (2017).

[66] GLOROT, X., AND BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth Int'l. Conf. on artificial intelligence and statistics* (2010), pp. 249–256.

[67] GOLDSTEIN, M., AND UCHIDA, S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one 11*, 4 (2016).

[68] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., AND BENGIO, Y. *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[69] GOYAL, K., AND SINGHAI, J. Review of background subtraction methods using Gaussian mixture model for video surveillance systems. *Artificial Intelligence Review 50*, 2 (2018), 241–259.

[70] GUBBI, J., BUYYA, R., MARUSIC, S., AND PALANISWAMI, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems 29*, 7 (2013), 1645–1660.

[71] HAQUE, A., KHAN, L., BARON, M., THURAISINGHAM, B., AND AGGARWAL, C. Efficient handling of concept drift and concept evolution over stream data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)* (2016), IEEE, pp. 481–492.

[72] HAQUE, M. A., AND MINENO, H. Proposal of Online Outlier Detection in Sensor Data Using Kernel Density Estimation. In *6th Int'l. Congress on Advanced Applied Informatics* (2017), IEEE, pp. 1051–1052.

[73] HARDY, C., LE MERRER, E., AND SERICOLA, B. MD-GAN: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2019), IEEE, pp. 866–877.

[74] HASENCLEVER, L., WEBB, S., LIENART, T., VOLLMER, S., LAKSHMINARAYANAN, B., BLUNDELL, C., AND TEH, Y. W. Distributed Bayesian learning with stochastic natural gradient expectation propagation and the posterior server. *The Journal of Machine Learning Research 18*, 1 (2017), 3744–3780.

[75] HERNÁNDEZ-LOBATO, J. M., AND ADAMS, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning* (2015), PMLR, pp. 1861–1869.

[76] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[77] HONG, L. Artificial immune system for anomaly detection. In *2008 IEEE international symposium on knowledge acquisition and modeling workshop* (2008), IEEE, pp. 340–343.

[78] HU, R., DELANY, S., AND MACNAMEE, B. Sampling with confidence: using k-NN confidence measures in active learning. In *Conf. papers* (2009), p. 50.

[79] HUANG, S.-Y., YU, F., TSAIH, R.-H., AND HUANG, Y. Network-traffic anomaly detection with incremental majority learning. In *Int'l. Joint Conf. on Neural Networks* (2015), IEEE, pp. 1–8.

[80] HUGHES-OLIVER, J. M. Population and empirical PR curves for assessment of ranking algorithms. *arXiv preprint arXiv:1810.08635* (2018).

[81] HUSMEIER, D. Random vector functional link (RVFL) networks. In *Neural Networks for Conditional Probability Estimation*. Springer, 1999, pp. 87–97.

[82] ILYAS, A., SANTURKAR, S., TSIPRAS, D., ENGSTROM, L., TRAN, B., AND MADRY, A. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175* (2019).

[83] INGRE, B., AND YADAV, A. Performance analysis of NSL-KDD dataset using ANN. In *Signal Processing And Communication Engineering Systems (SPACES), 2015 Int'l. Conf. on* (2015), IEEE, pp. 92–96.

[84] INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D., HEIDEMANN, J., AND SILVA, F. Directed diffusion for wireless sensor networking. *IEEE/ACM transactions on networking 11*, 1 (2003), 2–16.

[85] JAVAID, A., NIYAZ, Q., SUN, W., AND ALAM, M. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI Int'l. Conf. on Bio-inspired Information and Communications Technologies (formerly BIONETICS)* (2016), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 21–26.

[86] JIA, Q., GUO, L., JIN, Z., AND FANG, Y. Preserving model privacy for machine learning in distributed systems. *IEEE Transactions on Parallel and Distributed Systems 29*, 8 (2018), 1808–1822.

[87] KAEWTRAKULPONG, P., AND BOWDEN, R. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*. Springer, 2002, pp. 135–144.

[88] KALMAN, R. E., AND OTHERS. A new approach to linear filtering and prediction problems. *Journal of basic Engineering 82*, 1 (1960), 35–45.

[89] KAWACHI, Y., KOIZUMI, Y., AND HARADA, N. Complementary Set Variational Autoencoder for Supervised Anomaly Detection. In *Int'l. Conf. on Acoustics,Speech and Signal Processing* (2018), IEEE, pp. 2366–2370.

[90] KDDCUP-1999. http://kdd.ics.uci.edu/databases/kddcup99/kdcup99.html.

[91] KEMKER, R., AND KANAN, C. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563* (2017).

[92] KHALASTCHI, E., KALECH, M., KAMINKA, G. A., AND LIN, R. Online data-driven anomaly detection in autonomous robots. *Knowledge and Information Systems 43*, 3 (2015), 657–688.

[93] KIM, T., KANG, B., RHO, M., SEZER, S., AND IM, E. G. A multimodal deep learning method for Android malware detection using various features. *IEEE Trans. on Information Forensics and Security 14*, 3 (2018), 773–788.

[94] KINGMA, D. P., SALIMANS, T., AND WELLING, M. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems* (2015), pp. 2575–2583.

[95] KINGMA, D. P., AND WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[96] KIRKPATRICK, J., PASCANU, R., RABINOWITZ, N., VENESS, J., DESJARDINS, G., RUSU, A. A., MILAN, K., QUAN, J., RAMALHO, T., GRABSKA-BARWINSKA, A., ET AL. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences 114*, 13 (2017), 3521–3526.

[97] KOURTELLIS, N., KATEVAS, K., AND PERINO, D. FLaaS: Federated Learning as a Service. In *Proceedings of the 1st Workshop on Distributed Machine Learning* (2020), pp. 7–13.

[98] KUBE, N. Daniel Drescher: Blockchain basics: a non-technical introduction in 25 steps, 2018.

[99] KUCUKELBIR, A., TRAN, D., RANGANATH, R., GELMAN, A., AND BLEI, D. M. Automatic differentiation variational inference. *The Journal of Machine Learning Research 18*, 1 (2017), 430–474.

[100] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *The annals of mathematical statistics 22*, 1 (1951), 79–86.

[101] LAKHINA, A., CROVELLA, M., AND DIOT, C. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the*

4th ACM SIGCOMM Conf. on Internet measurement (2004), ACM, pp. 201–206.

[102] LAKHINA, A., CROVELLA, M., AND DIOT, C. Diagnosing network-wide traffic anomalies. In *SIGCOMM computer communication review* (2004), vol. 34, ACM, pp. 219–230.

[103] LI, Q., LIU, Y., LIU, Z., ZHANG, P., AND PANG, C. Efficient Forwarding Anomaly Detection in Software-Defined Networks. *IEEE Transactions on Parallel and Distributed Systems* (2021).

[104] LI, T., SAHU, A. K., TALWALKAR, A., AND SMITH, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine 37*, 3 (2020), 50–60.

[105] LIN, H.-T., LIN, C.-J., AND WENG, R. C. A note on Platt's probabilistic outputs for support vector machines. *Machine learning 68*, 3 (2007), 267–276.

[106] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision* (2014), Springer, pp. 740–755.

[107] LIU, B., DING, Z., AND LV, C. Distributed Training for Multi-Layer Neural Networks by Consensus. *IEEE transactions on neural networks and learning systems* (2019).

[108] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., AND BERG, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision* (2016), Springer, pp. 21–37.

[109] LIU, Y., GARG, S., NIE, J., ZHANG, Y., XIONG, Z., KANG, J., AND HOSSAIN, M. S. Deep anomaly detection for time-series data in

industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal* (2020).

[110] LUCAS, B. D., KANADE, T., ET AL. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia.

[111] LYU, L., BEZDEK, J. C., HE, X., AND JIN, J. Fog-embedded deep learning for the internet of things. *IEEE Transactions on Industrial Informatics 15*, 7 (2019), 4206–4215.

[112] LYU, L., JIN, J., RAJASEGARAR, S., HE, X., AND PALANISWAMI, M. Fog-empowered anomaly detection in IoT using hyperellipsoidal clustering. *IEEE Internet of Things Journal 4*, 5 (2017), 1174–1184.

[113] MACKAY, D. J. A practical Bayesian framework for backpropagation networks. *Neural computation 4*, 3 (1992), 448–472.

[114] MANN, H. B., AND WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.

[115] MAO, Y., YOU, C., ZHANG, J., HUANG, K., AND LETAIEF, K. B. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials 19*, 4 (2017), 2322–2358.

[116] MASSEY JR, F. J. The Kolmogorov-Smirnov test for goodness of fit. *Jrnl. of the American statistical Association 46*, 253 (1951), 68–78.

[117] MCMAHAN, B., MOORE, E., RAMAGE, D., HAMPSON, S., AND Y ARCAS, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics* (2017), PMLR, pp. 1273–1282.

[118] MIAO, X., LIU, Y., ZHAO, H., AND LI, C. Distributed online one-class support vector machine for anomaly detection over networks. *IEEE Trans. on cybernetics*, 99 (2018), 1–14.

[119] MINKA, T. Power ep. Tech. rep., Technical report, Microsoft Research, Cambridge, 2004.

[120] MINKA, T. P. Expectation propagation for approximate Bayesian inference. *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence* (2001).

[121] MOUSTAFA, N., CHOO, K.-K. R., RADWAN, I., AND CAMTEPE, S. Outlier Dirichlet Mixture Mechanism: Adversarial Statistical Learning for Anomaly Detection in the Fog. *IEEE Trans. on Information Forensics and Security* (2019).

[122] MOUSTAFA, N., AND SLAY, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Military communications and information sys. Conf.* (2015), IEEE, pp. 1–6.

[123] MOUSTAFA, N., AND SLAY, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Jrnl.: A Global Perspective 25*, 1-3 (2016), 18–31.

[124] MOŻEJKO, M., SUSIK, M., AND KARCZEWSKI, R. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861* (2018).

[125] NA, G. S., KIM, D., AND YU, H. DILOF: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM SIGKDD Int'l. Conf. on Knowledge Discovery & Data Mining* (2018), ACM, pp. 1993–2002.

[126] NAJAFABADI, M. M., VILLANUSTRE, F., KHOSHGOFTAAR, T. M., SELIYA, N., WALD, R., AND MUHAREMAGIC, E. Deep learning applications and challenges in big data analytics. *Jrnl. of Big Data 2*, 1 (2015), 1.

[127] NALISNICK, E., MATSUKAWA, A., TEH, Y. W., GORUR, D., AND LAKSHMINARAYANAN, B. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136* (2018).

[128] NAM, K., AND WANG, F. The performance of using an autoencoder for prediction and susceptibility assessment of landslides: A case study on landslides triggered by the 2018 Hokkaido Eastern Iburi earthquake in Japan. *Geoenvironmental Disasters 6*, 1 (2019), 19.

[129] NAPHADE, M., WANG, S., ANASTASIU, D. C., TANG, Z., CHANG, M.-C., YANG, X., YAO, Y., ZHENG, L., CHAKRABORTY, P., LOPEZ, C. E., SHARMA, A., FENG, Q., ABLAVSKY, V., AND SCLAROFF, S. The 5th AI City Challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (June 2021).

[130] NASEER, S., SALEEM, Y., KHALID, S., BASHIR, M. K., HAN, J., IQBAL, M. M., AND HAN, K. Enhanced network anomaly detection based on deep neural networks. *IEEE Access 6* (2018), 48231–48246.

[131] NAVARRO-ORTIZ, J., ROMERO-DIAZ, P., SENDRA, S., AMEIGEIRAS, P., RAMOS-MUNOZ, J. J., AND LOPEZ-SOLER, J. M. A survey on 5G usage scenarios and traffic models. *IEEE Communications Surveys & Tutorials 22*, 2 (2020), 905–929.

[132] NEAL, R. M. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo 2*, 11 (2011), 2.

[133] NEAL, R. M. *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012.

[134] NEISWANGER, W., WANG, C., AND XING, E. Asymptotically exact, embarrassingly parallel MCMC. *arXiv preprint arXiv:1311.4780* (2013).

[135] NEVAT, I., DIVAKARAN, D. M., NAGARAJAN, S. G., ZHANG, P., SU, L., KO, L. L., AND THING, V. L. Anomaly detection and attribution in networks with temporally correlated traffic. *IEEE/ACM Trans. on Networking 26*, 1 (2018), 131–144.

[136] NGUYEN, Q. P., LIM, K. W., DIVAKARAN, D. M., LOW, K. H., AND CHAN, M. C. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. *arXiv preprint arXiv:1903.06661* (2019).

[137] NGUYEN, T. D., MARCHAL, S., MIETTINEN, M., FEREIDOONI, H., ASOKAN, N., AND SADEGHI, A.-R. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (2019), IEEE, pp. 756–767.

[138] NICOLAU, M., MCDERMOTT, J., ET AL. A hybrid autoencoder and density estimation model for anomaly detection. In *Int'l. Conf. on Parallel Problem Solving from Nature* (2016), Springer, pp. 717–726.

[139] ODIATHEVAR, M. KiwiNet Emerging Innovator Award 2020, https://kiwinet.org.nz/EmergingInnovatorProgramme, Apr 2020.

[140] ODIATHEVAR, M., CAMERON, D., SEAH, W. K. G., FREAN, M., AND VALERA, A. Humans Learning from Machines: Data Science Meets Network Management. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), Invited Paper* (2021), IEEE, pp. 421–428.

[141] ODIATHEVAR, M., SEAH, W. K. G., AND FREAN, M. A Hybrid Online Offline System for Network Anomaly Detection. In *28th Int'l. Conf. on Computer Communication and Networks* (2019), IEEE, pp. 1–9.

[142] ODIATHEVAR, M., SEAH, W. K. G., FREAN, M., AND VALERA, A. An Online Offline Framework for Anomaly Scoring and Detecting New Traffic in Network Streams. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[143] ODIATHEVAR, M., SEAH, W. K. G., FREAN, M., AND VALERA, A. Patent on Hybrid Machine Learning System, Australian IP application number: 2019900788, https://www.ipaustralia.gov.au/patents.

[144] ODIATHEVAR, M., SEAH, W. K. G., FREAN, M., AND VALERA, A. Patent on Hybrid Machine Learning System and Method, World Intellectual Property Organization: Application number: PCT/NZ2020/050025, https://patentscope.wipo.int/search/en/search.jsf.

[145] OLUPS, R. *Zabbix 1.8 network monitoring*. Packt Publishing Ltd, 2010.

[146] OPPER, M., AND WINTHER, O. Gaussian processes for classification: Mean-field algorithms. *Neural computation 12*, 11 (2000), 2655–2684.

[147] O'REILLY, C., GLUHAK, A., AND IMRAN, M. A. Distributed anomaly detection using minimum volume elliptical principal component analysis. *IEEE Transactions on Knowledge and Data Engineering 28*, 9 (2016), 2320–2333.

[148] PERI, N., KHORRAMSHAHI, P., RAMBHATLA, S. S., SHENOY, V., RAWAT, S., CHEN, J.-C., AND CHELLAPPA, R. Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), pp. 622–623.

[149] PFENDER, J., VALERA, A., AND SEAH, W. K. G. Easy as ABC: A lightweight centrality-based caching strategy for information-centric IoT. In *Proceedings of the 6th ACM conference on information-centric networking* (2019), pp. 100–111.

[150] PLASTIRAS, G., TERZI, M., KYRKOU, C., AND THEOCHARIDCS, T. Edge intelligence: Challenges and opportunities of near-sensor machine learning applications. In *2018 ieee 29th international conference on application-specific systems, architectures and processors (asap)* (2018), IEEE, pp. 1–7.

[151] POKRAJAC, D., LAZAREVIC, A., AND LATECKI, L. J. Incremental local outlier detection for data streams. In *Symp. on computational intelligence and data mining* (2007), IEEE, pp. 504–515.

[152] PRECHELT, L. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[153] PRESS, W. H., WILLIAM, H., TEUKOLSKY, S. A., VETTERLING, W. T., SAUL, A., AND FLANNERY, B. P. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[154] RASCHKA, S., AND MIRJALILI, V. *Python Machine Learning, 2nd Ed.* Packt Publishing, Birmingham, UK, 2017.

[155] REKHTER, Y., LI, T., HARES, S., ET AL. A border gateway protocol 4 (BGP-4), 1994.

[156] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* (2015).

[157] RESEARCH EDUCATION ADVANCED NETWORK NEW ZEALAND. https://www.reannz.co.nz/.

[158] RIFAI, S., GLOROT, X., BENGIO, Y., AND VINCENT, P. Adding noise to the input of a model trained with a regularized objective. *CoRR abs/1104.3250* (2011).

[159] RIPE, N. Youtube hijacking a ripe ncc ris case study. *https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study* (2008).

[160] SAFARINEJADIAN, B., AND ESTAHBANATI, M. E. Consensus filter-based distributed variational Bayesian algorithm for flow and speed density prediction with distributed traffic sensors. *IEEE Systems Journal 11*, 4 (2015), 2939–2948.

[161] SALEHI, M., LECKIE, C., BEZDEK, J. C., VAITHIANATHAN, T., AND ZHANG, X. Fast memory efficient local outlier detection in data streams. *IEEE Trans. on Knowledge and Data Engineering 28*, 12 (2016), 3246–3260.

[162] SANTOS, M. R., ANDRADE, R. M., GOMES, D. G., AND CALLADO, A. C. An efficient approach for device identification and traffic classification in IoT ecosystems. In *Symp. on Computers and Communications* (2018), IEEE, pp. 00304–00309.

[163] SCARDAPANE, S., WANG, D., PANELLA, M., AND UNCINI, A. Distributed learning for random vector functional-link networks. *Information Sciences 301* (2015), 271–284.

[164] SCARDAPANE, S., WANG, D., AND UNCINI, A. Bayesian random vector functional-link networks for robust data modeling. *IEEE transactions on cybernetics 48*, 7 (2017), 2049–2059.

[165] SCHÖLKOPF, B., WILLIAMSON, R. C., SMOLA, A. J., SHAWE-TAYLOR, J., PLATT, J. C., ET AL. Support vector method for novelty detection. In *NIPS* (1999), vol. 12, Citeseer, pp. 582–588.

[166] SEEGER, M. Expectation propagation for exponential families. Tech. rep., 2005.

[167] SHAHID, M. R., BLANC, G., ZHANG, Z., AND DEBAR, H. IoT Devices Recognition Through Network Traffic Analysis. In *Int'l. Conf. on Big Data* (2018), IEEE, pp. 5187–5192.

[168] SHEN, K., JING, Z., AND DONG, P. A consensus nonlinear filter with measurement uncertainty in distributed sensor networks. *IEEE Signal Processing Letters 24*, 11 (2017), 1631–1635.

[169] SHI, S., CHU, X., AND LI, B. MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (2019), IEEE, pp. 172–180.

[170] SHI, Y., YANG, K., JIANG, T., ZHANG, J., AND LETAIEF, K. B. Communication-efficient edge AI: Algorithms and systems. *IEEE Communications Surveys & Tutorials 22*, 4 (2020), 2167–2191.

[171] SHIER, R. Statistics: 2.3 The Mann-Whitney U Test. *Mathematics Learning Support Centre. Last accessed 15* (2004), 2013.

[172] SILLITO, R. R., AND FISHER, R. B. Incremental one-class learning with bounded computational complexity. In *Int'l. Conf. on Artificial Neural Networks* (2007), Springer, pp. 58–67.

[173] ŠMÍDL, V., BÍM, J., AND PEVNÝ, T. Anomaly scores for generative models. *arXiv preprint arXiv:1905.11890*.

[174] SMITH, V., CHIANG, C.-K., SANJABI, M., AND TALWALKAR, A. Federated multi-task learning. *arXiv preprint arXiv:1705.10467* (2017).

[175] SOLAR WINDS. https://www.solarwinds.com/.

[176] SOMMER, R., AND PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symp. on Security and Privacy* (2010), IEEE, pp. 305–316.

[177] SPRINGENBERG, J. T., KLEIN, A., FALKNER, S., AND HUTTER, F. Bayesian optimization with robust Bayesian neural networks. In *Advances in neural information processing systems* (2016), pp. 4134–4142.

[178] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[179] SU, M.-Y. Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications 38*, 4 (2011), 3492–3498.

[180] SU, M.-Y. Using clustering to improve the KNN-based classifiers for online anomaly network traffic identification. *Jrnl. of Network and Computer Applications 34*, 2 (2011), 722–730.

[181] SULTANI, W., CHEN, C., AND SHAH, M. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 6479–6488.

[182] SUN, C., LV, K., HU, C., AND XIE, H. A Double-Layer Detection and Classification Approach for Network Attacks. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)* (2018), IEEE, pp. 1–8.

[183] TEERAPITTAYANON, S., MCDANEL, B., AND KUNG, H.-T. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), IEEE, pp. 328–339.

[184] TISSERA, M. D., AND MCDONNELL, M. D. Deep extreme learning machines: supervised autoencoding architecture for classification. *Neurocomputing 174* (2016), 42–49.

[185] TRAN, A. T. Network anomaly detection. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM) Focal Topic: Advanced Persistent Threats 55* (2017).

[186] TSOU, Y.-L., CHU, H.-M., LI, C., AND YANG, S.-W. Robust Distributed Anomaly Detection Using Optimal Weighted One-Class Random Forests. In *2018 IEEE International Conference on Data Mining (ICDM)* (2018), IEEE, pp. 1272–1277.

[187] UNIVERSITY OF OREGON: ROUTE VIEWS PROJECT. http://www.routeviews.org/routeviews/, 1997.

[188] VAN DE VEN, G. M., AND TOLIAS, A. S. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635* (2018).

[189] VAN STEIN, B., VAN LEEUWEN, M., AND BÄCK, T. Local subspace-based outlier detection using global neighbourhoods. In *Int'l. Conf. on Big Data* (2016), IEEE, pp. 1136–1142.

[190] VEHTARI, A., GELMAN, A., SIVULA, T., JYLÄNKI, P., TRAN, D., SAHAI, S., BLOMSTEDT, P., CUNNINGHAM, J. P., SCHIMINOVICH, D., AND ROBERT, C. P. Expectation Propagation as a Way of Life: A Framework for Bayesian Inference on Partitioned Data. *Journal of Machine Learning Research 21*, 17 (2020), 1–53.

[191] VINCENT, P., LAROCHELLE, H., BENGIO, Y., AND MANZAGOL, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th Int'l. Conf. on Machine learning* (2008), ACM, pp. 1096–1103.

[192] VIVENCIO, D. P., HRUSCHKA, E. R., DO CARMO NICOLETTI, M., DOS SANTOS, E. B., AND GALVAO, S. D. Feature-weighted k-nearest neighbor classifier. In *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symp. on* (2007), IEEE, pp. 481–486.

[193] WAN, M., SHANG, W., AND ZENG, P. Double behavior characteristics for one-class classification anomaly detection in networked control systems. *IEEE Trans. on Information Forensics and Security 12*, 12 (2017), 3011–3023.

[194] WANG, S., TUOR, T., SALONIDIS, T., LEUNG, K. K., MAKAYA, C., HE, T., AND CHAN, K. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (2018), IEEE, pp. 63–71.

[195] WANG, W., LIU, J., PITSILIS, G., AND ZHANG, X. Abstracting massive data for lightweight intrusion detection in computer networks. *Information Sciences 433* (2018), 417–430.

[196] WANG, X., HAN, Y., LEUNG, V. C., NIYATO, D., YAN, X., AND CHEN, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials 22*, 2 (2020), 869–904.

[197] WANG, X., JIN, B., DU, Y., CUI, P., AND YANG, Y. One-Class Graph Neural Networks for Anomaly Detection in Attributed Networks. *arXiv preprint arXiv:2002.09594* (2020).

[198] WASSERMAN, L. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[199] WASSERMAN, S., FAUST, K., ET AL. Social network analysis: Methods and applications.

[200] WELCH, B. L. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika 34*, 1/2 (1947), 28–35.

[201] WELLING, M., AND TEH, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), pp. 681–688.

[202] WOJKE, N., BEWLEY, A., AND PAULUS, D. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)* (2017), IEEE, pp. 3645–3649.

[203] WU, S. X., WAI, H.-T., LI, L., AND SCAGLIONE, A. A review of distributed algorithms for principal component analysis. *Proceedings of the IEEE 106*, 8 (2018), 1321–1340.

[204] XIE, K., LI, X., WANG, X., CAO, J., XIE, G., WEN, J., ZHANG, D., AND QIN, Z. On-line anomaly detection with high accuracy. *IEEE/ACM Trans. on networking 26*, 3 (2018), 1222–1235.

[205] XIE, M., HU, J., AND GUO, S. Segment-based anomaly detection with approximated sample covariance matrix in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 26*, 2 (2014), 574–583.

[206] XIE, M., HU, J., GUO, S., AND ZOMAYA, A. Y. Distributed segment-based anomaly detection with Kullback–Leibler divergence in wireless sensor networks. *IEEE Trans. on Information Forensics and Security 12*, 1 (2016), 101–110.

[207] XU, ET AL. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the World Wide Web Conf.* (2018), pp. 187–196.

[208] XU, B., CHEN, S., ZHANG, H., AND WU, T. Incremental k-NN SVM method in intrusion detection. In *8th Int'l. Conf. on Software Engineering and Service Science* (2017), IEEE, pp. 712–717.

[209] XU, D., LI, T., LI, Y., SU, X., TARKOMA, S., JIANG, T., CROWCROFT, J., AND HUI, P. Edge Intelligence: Architectures, Challenges, and Applications. *arXiv e-prints* (2020), arXiv–2003.

[210] XU, K., HU, W., LESKOVEC, J., AND JEGELKA, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[211] XU, M., LAKSHMINARAYANAN, B., TEH, Y. W., ZHU, J., AND ZHANG, B. Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems* (2014), pp. 3356–3364.

[212] YAN, Q., YU, F. R., GONG, Q., AND LI, J. Software-defined networking and distributed denial of service attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials 18*, 1 (2015), 602–622.

[213] YIN, C., ZHU, Y., FEI, J., AND HE, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access 5* (2017), 21954–21961.

[214] ZEILER, M. D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[215] ZHANG, S., BENENSON, R., OMRAN, M., HOSANG, J., AND SCHIELE, B. Towards reaching human performance in pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence 40*, 4 (2017), 973–986.

[216] ZHAO, Y., CHEN, J., WU, D., TENG, J., AND YU, S. Multi-task network anomaly detection using federated learning. In *Proceedings*

of the tenth international symposium on information and communication technology (2019), pp. 273–279.

Appendix A

Dataset Description

A.1 UNSW-NB15 dataset description

Table A.1 gives a breakdown of the UNSW-NB15 dataset. The list of features is given in Table A.2 and its description in Table A.3. More details are found in [122, 123].

Table A.1: UNSW-NB15: Number of flow records

Statistical features		**16 hours**	**15 hours**
Label	Normal	1,064,987	1,153,774
	Attack	22,215	299,068
Service	Others	650,635	597,790
	DNS	193,512	588,156
	HTTP	94,318	111,955
	FTP-Data	61,668	64,115
	SMTP	38,845	42,800
	FTP	24,861	24,229
	SSH	23,361	23,799

Table A.2: List of features in the UNSW-15 dataset

Type	**Features**
Nominal	state, service, protocol
Binary	is_sm_ips_ports, is_ftp_login
Numerical	dur, sbytes, dbytes, sttl, dttl, sloss, dloss, sload, dload, spkts, dpkts, swin, dwin, stcpb, dtspb, smeansz, dmeansz, response_body_len, sjit, djit, sintpkt, dintpkt, tcprtt, synack, ackdat, ct_state_ttl, ct_flw_http_mthd, ct_ftp_cmd, ct_srv_dst, ct_dst_src_ltm, trans_depth, ct_srv_src, ct_dst_ltm, ct_src_ ltm, ct_src_dport_ltm, ct_dst_sport_ltm

Table A.3: Description of features in the UNSW-15 dataset

state	The state and its dependent protocol, e.g. ACC, CLO
service	http, ftp, ssh, dns, ssl, .., else (-)
protocol	Transaction protocol: xnet, swipe, dcn, pnni, ...
is_sm_ips_ports	If source IP equals to destination IP addresses and port numbers are equal, this variable takes value 1 else 0
is_ftp_login	If the ftp session is accessed by user and password then 1 else 0.
dur	Record total duration
sbytes	Source to destination bytes
dbytes	Destination to source bytes
sttl	Source to destination time to live
dttl	Destination to source time to live
sloss	Source packets retransmitted or dropped
dloss	Destination packets retransmitted or dropped
sload	Source bits per second
dload	Destination bits per second
spkts	Source to destination packet count

dpkts	Destination to source packet count
swin	Source TCP window advertisement
dwin	Destination TCP window advertisement
stcpb	Source TCP sequence number
dtspb	Destination TCP sequence number
smeansz	Mean of the flow packet size transmitted by the src
dmeansz	Mean of the flow packet size transmitted by the dst
response_body _len	The content size of the data transferred from the server's http service.
sjit	Source jitter (mSec)
djit	Destination jitter (mSec)
sintpkt	Source inter-packet arrival time (mSec)
dintpkt	Destination inter-packet arrival time (mSec)
tcprtt	The sum of 'synack' and 'ackdat' of the TCP.
synack	The time between the SYN and the SYN_ACK packets of the TCP.
ackdat	The time between the SYN_ACK and the ACK packets of the TCP.
ct_state_ttl	No. for each state according to specific range of values for sttl and dttl.
ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service.
ct_ftp_cmd	No of flows that has a command in ftp session.
ct_srv_dst	No. of connections that contain the same service and destination address in 100 connections.
ct_dst_src_ltm	No of connections of the same src and the dst address in in 100 connections.
trans_depth	the depth into the connection of http request/response transaction.
ct_srv_src	No. of connections that contain the same service

	and src address in 100 connections.
ct_dst_ltm	No. of connections of the same dst address in 100 connections.
ct_src_ ltm	No. of connections of the same src address in 100 connections.
ct_src_dport_ltm	No. of connections of the same src address and the dst port in 100 connections.
ct_dst_sport_ltm	No. of connections of the same dst address and the src port in 100 connections.

A.2 CTU13 Datasets

Table A.4 gives a breakdown of the CTU13 datasets. The features in this dataset are duration, protocol, state, source type of service, destination type of service, total packets, total bytes and source bytes.

Table A.4: CTU13 Datasets

Id	Duration(hrs)	# Packets	# NetFlows	Size	Bot	# Bots
1	6.15	71,971,482	2,824,637	52 GB	Neris	1
3	66.85	167,730,395	4,710,639	121 GB	Rbot	1
9	5.18	115,415,321	2,753,885	94 GB	Neris	10
13	16.36	50,888,256	1,925,150	34 GB	Virut	1

A.3 NSLKDD

The training dataset was collected for seven weeks and the testing dataset was collected in the following 2 weeks. The testing data contains variations of the anomalies found in the training data to make the anomaly

detection process realistic as most novel attacks are variants of known attacks [43].

The dataset contains 41 features, of which three of them are nominal, six of them are binary and the remaining are numerical. Table A.5 shows the count of each class in the dataset. In this thesis, the labels of the attacks are replaced as anomaly.

Table A.5: NSLKDD 2009 dataset

	Total	**Normal**	**DoS**	**Probe**	**R2L**	**U2R**
Training set	125973	67343	45927	11656	995	52
Testing set	22544	9711	7458	2421	2754	200

Appendix B

Data Preprocessing, Model Training and Other Results

B.1 Hybrid Online Offline Framework

B.1.1 Data Preprocessing

Firstly, one-hot encoding of the nominal features is performed transforming them into binary features. In the UNSW-15 dataset, the one-hot encoded nominal features are *'state'* and *'service'*. The feature *'protocol'* is dropped because it has 135 different protocols from different network layers, including network, transport and application. All of the normal data is contained within seven of these and the rest are anomalous data. If these were included, the machine learning algorithm could learn the protocol names instead of numerical features to discriminate the data. This would lead to unnaturally good results but it would not validate the framework. Principal Component Analysis (PCA) is used to transform all of the binary features into numerical attributes. Here, PCA is not used for dimensionality reduction and so these components continue to explain 100% of the variability of these binary features. These features are then normalized using Min-Max scaling. More information about the features are available in

Appendix A.

For the rest of the numerical features which contain outliers that can distort the data, the method used by Su *et al.* [179] is used. It is given by

$$\text{Normalization of } f_i = \frac{1 - e^{-lf_i}}{1 + e^{-lf_i}}. \tag{B.1}$$

In this equation, f_i is the observed value of the feature i and l is a constant. Each numerical feature has its own l value. The constant l is determined such that the average of feature i of the normal data instances is mapped to 0.5.

The values in the latent layer of the AE need not be in the range of [0,1]. For training of the online model, robust-scaling is performed on the latent layer representation based on the values of the latent layer of the training data.

B.1.2 Model Training

The offline models are trained with five hidden layers as in [27, 51]. The latent layer dimension is determined based on a rule of thumb, $d' = [1 + \sqrt{d}]$ [27, 138]. The batch size for updating weights is fixed at 500 since the training set size is constant for all of the data sets and the number of epochs is varied. The Adadelta algorithm is utilised [214] and the weights are initialised using the uniform distribution [66]. The model which is trained for 200 epochs is used for comparison. ReLu activations are used in each hidden layer and Sigmoid in the output layer because the input values are normalized in the range [0,1]. Noise from the Gaussian distribution with mean 0 and standard deviation of 0.2 is added to the normalised input of the training data for the DAE.

All online models are trained on the latent layer representation of the offline models in batches of 1000 for consistency. In a real setting, this value largely depends on how much changes is expected in the network and computational capabilities. Using a larger value increases accuracy as

more data will be used. But it will result in longer waiting times. The online models can also be trained in time windows with similar consequence. The retraining criteria here is when the total number of data for retraining reaches 1000. For the OCSVM with the Gaussian kernel, if data is selected using thresholding and MW test, ν is set at 0.0001 ($< 1/1000$) because all data selected is expected to be normal data. If it is a purely online model, set $\nu = 0.5$. The γ value in OCSVM is set as the reciprocal of the number of features. In this case, $\gamma = 1/d'$. The number of components for GMM and the parameters are determined according to Bayesian Information Criterion (BIC) and expectation maximization. The number of neighbours for the LOF is taken to be 2% of the batch size. The bandwidth parameter for KDE is $\sqrt{d'/2}$ with respect to the number of features. For the IOCSVM, all of the support vectors are retained. For IGMM, 5% forgetting rate is used as in [1]. The purely online models are fit with all of the values in the previous batch and perform outlier detection as the data arrives in the current batch.

To remain consistent, the MW-test is also performed in batches of 1000. θ is determined using grid-search in the range of [50,100) by performing the MW-test on the REs of the validation set against those of the training set. Since both groups consist only of normal data, the highest value that gives us a p-value of 1.0 is taken because they are expected to be exact under the null hypothesis. For the UNSW-NB15 dataset, θ is calculated to be 85^{th} percentile. During testing, H_0 is rejected at $\alpha = 5\%$ significance at each iteration.

All experiments are implemented using Python 3.7.3 and run on GNU/Linux x86_64 with an 8xIntel Core i7-6700 CPU at 3.4 GHz, 15.5 GB RAM.

B.1.3 Training and Scoring Times

The offline models took 290.76s to train on average. Table B.1 shows the total time to score the test set of 119,000 records. This time includes on-

line training, the MW test and updating $thres$. The average training time shows the time taken for online training.

Table B.1: Training and scoring times based on UNSW-NB15

Online Models	**Total Scoring Time (s)**	**Average Training Time**
IOCSVM	85.49	0.00184
IGMM	328.90	4.170
OCSVM	86.29	0.000975
GMM	81.28	0.00766
LOF	106.41	0.00899
KDE	90.95	0.000725

IGMM takes a long time as similar clusters are merged, whereas GMM features complete re-training. The time difference of retaining support vectors in the IOCSVM *vs* not retaining in the OCSVM is negligible. The IOCSVM, OCSVM, GMM and KDE make better models in terms of quick training but IGMM could also work if the training were parallelized. We compare their results in the next sub section.

B.2 Signature-Based Hybrid Online Offline Framework

B.2.1 Data Preprocessing

One hot encoding of the nominal features are done to make them into binary features. Next, all binary features are decomposed into eight components using PCA. These eight components explain at least 82% of the variability of these binary features. Taking additional components accounts for marginal gains with respect to this dataset.

The feature *'num_outbound_cmds'* is dropped as it only takes the value 0 in the entire dataset. The features are normalized to the range [0,1]. For the eight PCA components, Min-Max scaling is used. Min-Max scaling takes each value and subtracts the minimum value and divides over the feature range. For the numerical features due to the effect of outliers which can highly distort the data, Equation B.1 is used.

Model Training

The weighted euclidean distance is used to calculate the radius for Rad-NN and the distance between each point.

$$D(\mathbf{a}, \mathbf{b}) = \sqrt{w_1(a_1 - b_1)^2 + \cdots + w_n(a_n - b_n)^2} \tag{B.2}$$

This is because some of the features have more predictive power and the Rad-NN does not learn that some features are more discriminative than others. The feature weights are computed based on Chi Squared statistic [192] and rescaled to the range [0, 1] using Min-Max scaling.

20% of the training set is used to determine the value for the radius. For each point in this set, the average of its distances from all other points is firs computed. Since the distribution is skewed, the 50th percentile of these distances is taken to be the radius. Figure B.1 shows the distribution. The smaller the radius, there is more responsibility on the online model for classification. The larger the radius, the system will obtain votes from points that are further away which is not appropriate.

The system contains a total of 6 parameters to specify. They are k from Equation 3.1, the threshold value $thres$ for the acceptance of the classification of Rad-NN, u for selecting the point for online training, the number of points for online training and the C and γ parameters of the RBF kernal. The training set is split into 80%-20% proportion. The Rad-NN is trained on the 80% of the training set and the remaining 20% is used as the validation dataset to find optimal values for these parameters. The value for k in Equation 3.1 and $thres$ are dependent. A larger k value gives a

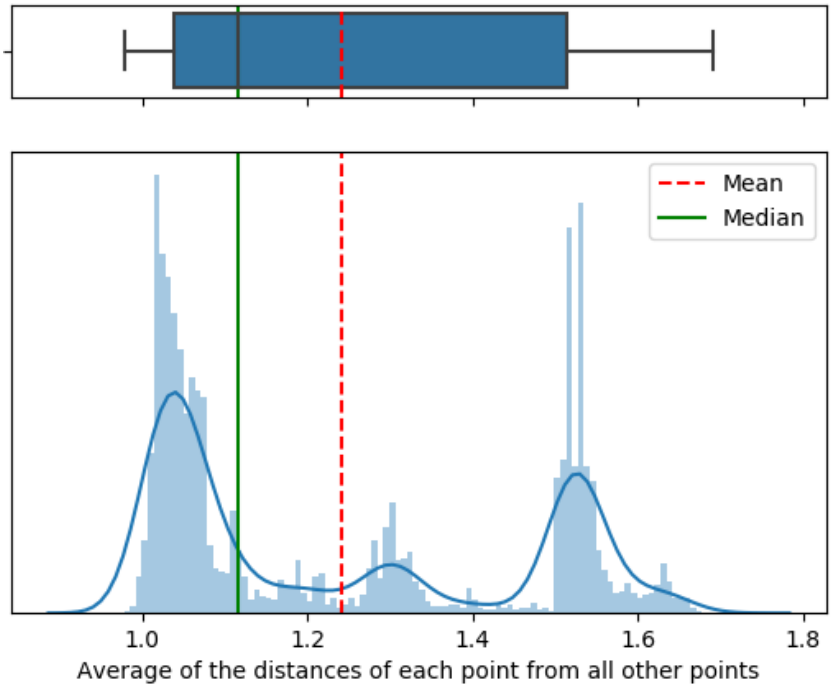
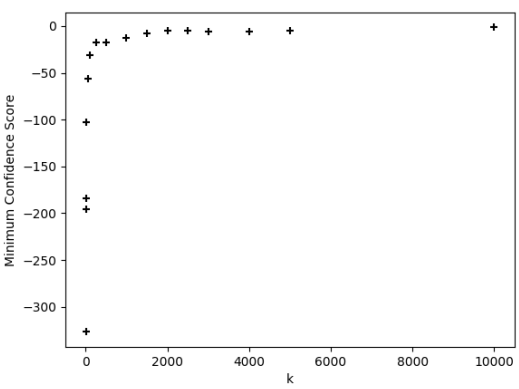
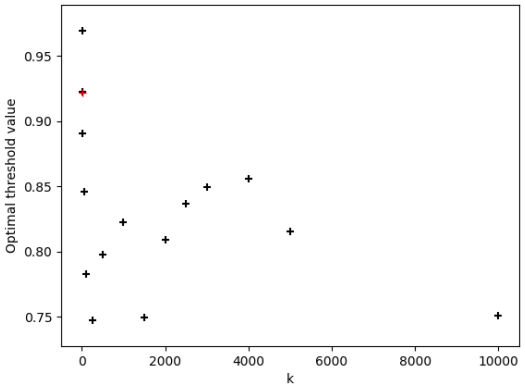


Figure B.1: Distribution of the average of the distances

more robust confidence score for C^*. Calculating the confidence scores of each label classified by the Rad-NN, Figure B.2a shows that the minimum value tends towards 0 as k increases. Taking a high value of k however, is computationally costly as it takes longer to calculate the confidence of the classified label and might not be feasible in real time. For each k, an optimal value for $thres$ is taken to be the maximum of the confidence scores of all the misclassified points in the validation set. In the experiment, we take $k = 10$ and the corresponding value for $thres$ is 92.2% as shown by the red point on Figure B.2b. Subsequently, u must be greater than $thres$ as it is the second layer threshold to obtain high confidence points for online training. Using grid-search method on the system, the following parameter values are obtained. u =96% and 1300 for the number of points for online training gives the highest detection rate on the validation set as seen in Figure B.2. Similarly, using grid-search, $\gamma = 0.1$ gave the highest detection rate



(a) Minimum C^* value against k



(b) Optimal *thres* value against k

while $C = 100$ gave the lowest false positive rate. If a regularised SVM with a balance between bias and variance was used, the number of support vectors to retain for subsequent training will be large. As such, more high confident points need to be used at the expense of longer waiting time or the SVM be retrained regularly to ensure concept-drift is captured. Figures B.3b and B.3a shows the results of the validation process. For a low values of C or for $\gamma > 1$, the number of SVMs trained is very high.

Before deployment in real systems, simulations would be done online prior to start up. To replicate this in the experiments, the online SVM is initialised using the first 1300 points in the testing set. The first 1300 points consist of 574 normal data, 453 DoS, 130 R2L, 130 Probe and 13 U2R points. The attacks are labelled as anomalies.

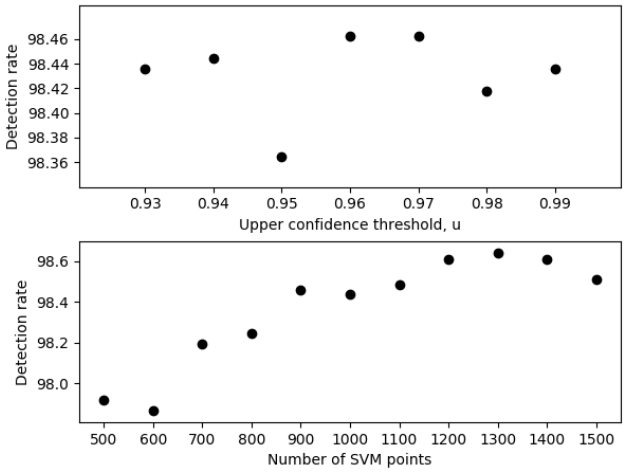
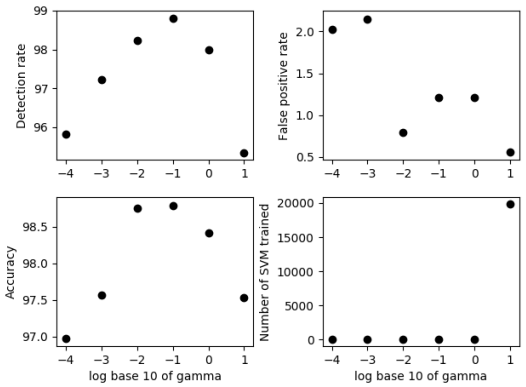
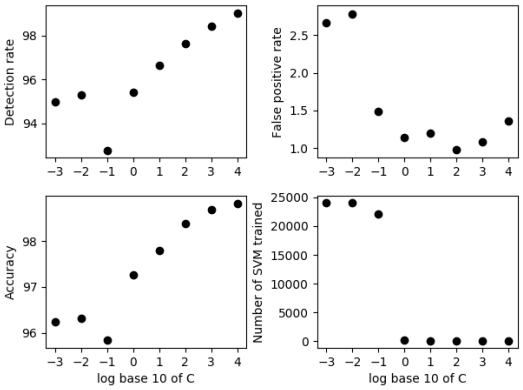


Figure B.2: Detection rate of validation set against u and number of points used in online training



(a) Validation results for γ in RBF kernel training



(b) Validation results for C in RBF kernel

Appendix C

Bayesian Inference and EP-BRVFL-AE

C.1 Bayesian Regression

Let $\{\mathbf{X}, \mathbf{y}\}$ denote the dataset where $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ is of size n and dimension d and $\mathbf{y} = [y_1, y_2, \ldots, y_n]$. Let $\mathbf{w}$ denote the parameters of a function $f : X \to \mathbb{R}$ to be determined. The general form is given by

$$y_i = f_{\mathbf{w}}(\mathbf{x_i}) + \epsilon_i, \tag{C.1}$$

where ϵ_i is white noise; drawn from a zero-centered Gaussian distribution with variance σ^2, $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2)$. Thus, $y_i \sim \mathcal{N}(f_{\mathbf{w}}(\mathbf{x_i}), \sigma^2)$ and the likelihood is given as

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{i=1}^{n} P(y_i|\mathbf{x_i}, \mathbf{w}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|y_i - f_{\mathbf{w}}(\mathbf{x_i})|^2}{2\sigma^2}\right) \tag{C.2}$$

An optimal value of $\mathbf{w}$ can be found by maximizing Equation C.2. This is known as the Maximum Likelihood Estimate (MLE). This method is not suitable for distributed training especially when the data at each site is different. Different 'optimal' values of $\mathbf{w}$ cannot be combined. By placing

a prior distribution on $\mathbf{w}$, $P(\mathbf{w})$, Bayes law is used to infer the posterior distribution:

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma^2) = \frac{P(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)P(\mathbf{w})}{P(\mathbf{y}|\mathbf{X})} \tag{C.3}$$

$$P(\mathbf{y}|\mathbf{X}) = \int P(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)P(\mathbf{w})d\mathbf{w} \tag{C.4}$$

where $P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma^2)$ is the posterior distribution, $P(\mathbf{w})$ is the prior distribution and $P(\mathbf{y}|\mathbf{X})$ is the normalizing constant or the marginal likelihood. A common choice for the prior is also a zero centered Gaussian, $P(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbf{S})$ where $\mathbf{S}$ is usually a diagonal covariance matrix. Again, if a point estimate suffices, the Maximum-A-Posteriori (MAP) estimate $\mathbf{w}_{MAP} = \text{argmax}_{\mathbf{w}} P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma^2)$ can be found using only the numerator of Equation C.3. However, a full Bayesian approach accounts for all posteriori models. Hence, for a new datapoint $\hat{\mathbf{x}}$, the posterior predictive distribution is found by integrating over the model parameters.

$$P(\hat{y}|\hat{\mathbf{x}}, \mathbf{y}, \mathbf{X}, \sigma^2) = \int_{\mathbf{w}} P(\hat{y}|\mathbf{w}, \hat{\mathbf{x}}, \mathbf{y}, \mathbf{X}, \sigma^2)P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma^2)d\mathbf{w} \tag{C.5}$$

For deep neural networks, computing the marginal likelihood becomes intractable and no closed form solutions are available. However, for Bayesian linear regression, with the conjugate prior distribution for the likelihood, closed form solution can be computed. Then the posterior distribution will be in the same form as the prior. For linear regression, $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$. Assume known noise variance σ^2 and the Gaussian prior for the weights $P(\mathbf{w}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$, the posterior distribution is also Gaussian with mean $\mathbf{m}$ and covariance matrix $\mathbf{\Sigma}$ as shown below.

$$\mathbf{m} = \mathbf{\Sigma}(\mathbf{S}_0^{-1}\mathbf{m}_0 + \frac{1}{\sigma^2}\mathbf{\Sigma}^{-1}\mathbf{X}\mathbf{y}) \tag{C.6}$$

$$\mathbf{\Sigma} = (\mathbf{S}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X})^{-1} \tag{C.7}$$

The posterior predictive distribution is also Gaussian and available in closed form. For more in depth formulation, readers are referred to [20].

$$P(\hat{y}|\hat{\mathbf{x}}, \mathbf{y}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{m}^T\hat{\mathbf{y}}, \phi(\hat{\mathbf{x}})^2) \tag{C.8}$$

$$\phi(\hat{\mathbf{x}})^2 = \sigma^2 + \hat{\mathbf{x}}^T\Sigma\hat{\mathbf{x}} \tag{C.9}$$

C.1.1 Exponential family of distributions

The probability density function of the distributions in the exponential family follow the following form.

$$P(x|\boldsymbol{\theta}) = h(x)\exp(\boldsymbol{\eta}(\boldsymbol{\theta})^T\mathbf{t}(x) - \mathbf{A}(\boldsymbol{\theta}))$$

where $\boldsymbol{\eta}(\boldsymbol{\theta})$ is the natural parameter of the distribution and $\mathbf{t}$ and $\mathbf{A}$ are vector valued functions. The Gaussian, Gamma, Inverse-Gamma, Exponential, Dirichlet, Chi-Squared, Wishart are some examples of distributions in the exponential family. For more details, readers can refer to [38].

For Inverse-Gamma(α, β), the mapping to natural parameters is given by

$$r = -\alpha - 1 \qquad Q = -\beta$$

C.1.2 Gaussian distribution with known mean and unknown variance

For $P(y|\mu, \sigma^2) = N(y|\mu, \sigma^2)$ where μ is known and σ^2 is unknown, the likelihood for a vector y of n independent and identically distributed observations is

$$P(y|\sigma^2) \propto \sigma^{-n}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2\right)$$

The corresponding conjugate prior density is an Inverse-Gamma distribution,

$$P(\sigma^2) \propto (\sigma^2)^{-(\alpha+1)}\exp(-\beta/\sigma^2)$$

with hyper-parameters α, β. The resulting posterior density is

$$\begin{aligned} P(\sigma^2|y) &\propto P(y|\sigma^2)P(\sigma^2) \\ &= (\sigma^2)^{-(\alpha+n+1)} \exp\left(-\frac{2\beta + \sum_{i=1}^{n}(y_i - \mu)^2}{2\sigma^2}\right). \end{aligned}$$

This posterior follows an Inverse-Gamma distribution:

$$P(\sigma^2|y) \propto \text{IG}\left(n + \alpha, -\frac{2\beta + \sum_{i=1}^{n}(y_i - \mu)^2}{2}\right)$$

The estimate for the precision parameter follows from the fact that if γ follows Gamma(α, β) then $1/\gamma$ follows Inverse-Gamma(α, β). The mode of the Inverse-Gamma distribution with parameters α, β is $\beta/(\alpha + 1)$.

C.2 EP-BRVFL-AE

Algorithm 5 Preprocessing for EP-BRVFL-AE(C)

1: **for** $k = 1, \ldots, K$ **do**
2: **At edges**: Send $\bar{f}_j^k$ to central
3: **end for**
4: **At central**: $\bar{f}_j = \frac{1}{K}\sum_{k=1}^{K} \bar{f}_j^k$
5: **At central**: Compute l_j and send to edges
6: **At edges**: Normalize data using (B.1)

Firstly, before any training can occur, data from all of the sites need to be normalised appropriately for which Equation B.1 is used.

In EP-BRVFL-AE(D), a broadcast of $\bar{f}_j^k$ to all sites is required. This incurs an additional communication cost of d and computation cost of Kn_k. Alternatively, each site can use its own data to perform normalisation especially if the data distribution is similar between the sites.

The middle layer of the BRVFL-AE is ζ times the input layer d, and $\zeta = 10$ for comparison unless otherwise mentioned. The algorithm is im-

plemented with $\delta_0 = 1$. The number of edge sites is fixed to 10 for comparison and the data is randomly partitioned unless specified. The hyper-parameters are the MAP estimates of the hyper-posteriors determined by the EP. The initial hyper-prior parameters are such that the initial estimates are $\sigma^2 = 0.01$ and $\gamma = 0.01$ as in [164].

All experiments are implemented using Python 3.8.2 and run on GNU/Linux x86_64 with an Intel Core i7-7567U CPU at 3.5 GHz, 16 GB RAM.

C.3 Determining Hyper-parameters

C.3.1 Averaging Estimates

This method is an alternative to obtaining the values for hyper-parameters σ^2 and γ. It has a lower communication and computation complexity but relies on some ad hoc averaging. This iterative procedure to obtain MAP estimates is described in [20, 164]. The updating equations are stated as follows:

$$\gamma = \frac{\eta + 2\alpha_\gamma}{\|\mathbf{m}\|_2^2 + 2\beta_\gamma} \tag{C.10}$$

$$\sigma^2 = \frac{\|X - \mathbf{H}\mathbf{w}\|_2^2 + \beta_\sigma}{n - \eta + 2\alpha_\sigma} \tag{C.11}$$

where η is

$$\eta = \sum_{l=1}^{B} \frac{\lambda_l}{\gamma + \lambda_l} \tag{C.12}$$

and λ_l are eigenvalues of $(1/\sigma^2)\mathbf{H}^T\mathbf{H}$.

The iterative method begins with a fixed value for σ^2 and γ and is updated iteratively at the central location before being sent to the distributed sites. It involves placing Algorithm 3 into an iterative loop. Let n_k denote the number of data at site k. The following steps in Algorithm 6 are added to Algorithm 3. Algorithm 6 can be easily adjusted for a fully distributed version by using only values from neighbouring sites.

Algorithm 6 Iterative Averaging for Hyper-parameters

1: Initialise values for $\alpha_\sigma, \beta_\sigma, \alpha_\gamma, \beta_\gamma, \sigma^2$ and γ.

At each site:

2: Calculate $\|X_k - \mathbf{H}_k\mathbf{w}\|_2^2$ and η_k using (C.12) and send them and n_k to central.

At central:

3: Calculate $n = \sum_{k=1}^{K} n_k$ and $\eta = \frac{1}{K}\sum_{k=1}^{K} \eta_k$

4: and TSE $= \sum_{k=1}^{K} \|X_k - \mathbf{H}_k\mathbf{w}\|_2^2$

5: Update γ and σ^2 using (C.10), (C.11)

6: Send updated γ and σ^2 to sites.

The eigenvalues can be calculated once for $\mathbf{H_k}^T\mathbf{H_k}$, stored in memory and updated with each new σ^2. This approach automatically calculates the MAP estimates for γ and σ^2. It also considers the eigenvalues which corresponds to the curvature of the likelihood function [20].

To estimate γ and σ^2 using this iterative averaging algorithm, computational cost at each site is $B^2d + n_kd$ and $B^3 + B$. The communication cost at the site is $3c$. At the central the computation cost is $3K + 2$ and communication cost is $2c$. These cost are also added to the sites in the fully distributed implementation.