Interpretability and Performance of Deep Neural Network based Anomaly Detection in Cyber Security and Telecommunications

Author: Ashima Chawla

Supervised by:

Dr. Paul Jacob and Dr. Sheila Fallon

Technological University of the Shannon: Midlands Midwest

Ollscoil Teicneolaíochta na Sionainne:
Lár Tíre Iarthar Láir

A Thesis Submitted for the degree of

Doctor of Philosophy

June, 2022

I dedicate my dissertation work to my family and many friends.

Abstract

The rapid development of technology and proliferation of data have driven businesses to pursue anomaly detection research. The application of artificial neural networks (ANNs) in anomaly detection achieves the state-of-the-art, but the end user cannot easily interpret their output. Therefore, to leverage ANNs in the field of Anomaly Detection, it is important to interpret the neural network models. This thesis addresses the question of whether it is possible to design and develop high performance and interpretable anomaly detection solutions based on artificial neural networks.

Anomaly detection is an important technique in Cyber Security as, compared to signature based methods, an anomaly detection based approach is capable of detecting previously unseen attacks. One approach to develop a Host Based Intrusion Detection System for Cyber Security is to examine sequences of traces of operating system calls. Two approaches to anomaly detection for sequential data are a prediction based approach and a reconstruction error based approach. A prediction based approach predicts the next element in a sequence based on the previously observed sequence. The work incorporates stacked Convolutional Neural Network (CNNs) with Gated Recurrent Units (GRUs) to analyse the operation system call sequences with an order of magnitude smaller training times. The reconstruction error based approach leverage bidirectional autoencoders to detect the anomalous system call sequences. This approach achieved better Area Under the Curve (AUC) when compared to the predictive approach. This approach to anomaly detection forms the basis for an interpretability framework.

Anomaly Detection is also an important technique in telecommunications monitoring. The Cluster Characterized Autoencoder (CCA) Framework was designed, implemented, and evaluated to identify candidate anomalies and interpret the model predictions. This framework addresses the neural network interpretability to support network engineers to perform troubleshooting and aid in root cause analysis.

Keywords: Anomaly Detection, Interpretation, System call traces

Acknowledgements

This research has received funding from Irish Research Council Enterprise Partnership Scheme Postgraduate Scholarship under project EBPPG/2019/76 and the European Union Horizon 2020 research and innovation programme under grant agreement No. 700071 for the PROTECTIVE project. I would like to express my deep gratitude to Dr. Paul Jacob, Dr. Brian Lee, Dr. Sheila Fallon, and Dr. Enda Fallon of Technological University of the Shannon: Midlands Midwest for their consistent support, mentorship in conducting this research. Your feedback and experience has been invaluable throughout the course of this work. Really, could not have asked for a better supervisor.

A special mention of thanks to my friends in SRI and my industrial mentors Jimmy O' Meara, Dr. Saman Fegghi, Dr. Erik Aumayr, Devashish Rughwani and Paddy Farrell for their constant support and cooperation during my research. Their timely help and friendship will always be remembered.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my supporting partner Pradeep Babu for the continuous love and support without whom this amazing PhD journey would not have been possible. I thank everyone for putting up with me at difficult moments when I felt stumped and for guiding me on to follow my dream of getting this degree. This would not have been possible without your unwavering and unselfish love and support for me at all times.

Finally, I wish to owe everything to my beautiful family for their mental support throughout this research work.

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Doctor of Philosophy program in the Department of Electronic, Computer and Software Engineering at Technological University of the Shannon: Midlands Midwest is entirely my own work and has not been taken from the work of others and to the extent that such work has been cited and acknowledged within the text of my work.

I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this Institute.
- Where any part of this thesis has been previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the others, the source is always given.
- With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Ashima Chawla

Contents

Figures and Tables

Figures

Tables

Nomenclature

1G	1st Generation
4G	4th Generation
5G	5th Generation
AD	Anomaly Detection
ADFA-LD	Australian Defence Force Academy - Linux Dataset
ADS	Anomaly Detection System
AE	Autoencoder
ANNs	Artificial Neural Networks
ARIMA	Autoregressive Moving
AUC	Area Under Curve
AUC-PR	Area Under Precision Recall
BiRNN	Bidirectional Neural Network
BN	Batch Normalization
BPTT	BackPropogation Through Time
CADAMANT	Context Anomaly Detection for Maintenance and Network Troubleshooting
CAGR	Compound Annual Growth Rate
CCA	Cluster Characterized Autoencoder
CNN	Convolutional Neural Network
CNN-LSTMED	Convolutional Neural Networks Long Short-Term Encoder-Decoder
CPU	Central Processing Unit
Crisp-DM	Cross Industry Standard Process for Data Mining
CUDA	Compute Unified Device Architecture
CuDNN	NVIDIA CUDA® Deep Neural Network library
DBO	Distance based Outlier
DeepLift	Evolved Packet Core
DIFFI	Depth-based Feature Importance for the Isolation Forest
DNN	Deep Neural Network
eNodeB	Evolved Node B

EPC	Evolved Packet Core
ERAB	E-UTRAN Radio Access Bearer
FP	False Positive
FPR	False Positive Rate
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HIDS	Host Intrusion Detection System
ID	Intrusion Detection
IDS	Intrusion Detection Systems
IF	Isolation Forest
IoT	Internet of Things
IQR	Interquartile Range
KDD	Knowledge Discovery in Database
KNN	K Nearest Neighbor
KPI	Key Performance Indicator
LIME	Local Interpretable Model-agnostic Explanations
LSTM	Long Short Term Memory
LTE	Long Term Evolution
MAC	Media Access Control Address
MAE	Mean Absolute Error
MCOD	Micro-Cluster based Outlier Detection
ML	Machine Learning
MMEs	Mobility Management Entity
MSE	Mean Squared Error
NIDS	Network Intrusion Detection System
NLP	Natural Language Processing
PCA	Principal Component Analysis
PDCCH	Physical Downlink Control Channel
PM	Performance Management
RAC_UE_REF	Radio Admission Control User Equipment Reference
RAN	Radio Access Network
RCA	Root Cause Analysis

RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RSRP	Reference Signal Received Power
SGW	Signaling Gateway
SHAP	SHapley Additive exPlanations
SINR	Signal to Interference & Noise Ratio
SSC-OMP	Sparse SubspaceClustering by Orthogonal Matching Pursuit
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
TP	True Positive
TPR	True Positive Rate
UE	User Equipment
XAI	eXplainable Artificial Intelligence

Chapter 1

Introduction

This thesis focuses on the design and development of Artificial Neural Networks (ANNs) based anomaly detection solutions in the field of cyber security and telecommunications. The thesis presents an intuitive interpretation of ANNs. The initial 18 months of research were based on Cyber Security in the TUS, Software Research Institute (SRI)[1]. This led to more focused research in the area of network Performance Management (PM) in 2019 [2].

1.1 Motivation

1.1.1 Anomaly Detection in Cyber Security

"Data is the new currency, and with open data, the possibilities are endless" (Gates & Matthews 2014). With advancements in today's era of technology and the exponential growth of data, security is becoming one of the major concerns for organizations due to the high level of attacks on organization networks and systems. Attack victims include banking, social media, healthcare, banks, cryptocurrency, cloud computing and the Internet of Things (IoT) ; where every aspect of applied science is being exploited by human hackers. Anomaly Detection has become one of the most challenging aspects of modern digital technology and it has become imperative to minimize and possibly avoid the impact of attacks on the system.

In recent years, rising ransomware attacks have contributed to increased momentum in security research. Enterprises rely on cybersecurity to protect their computing systems against unauthorised access. An Artificial Intelligence (AI) based anomaly detection system has the potential to combat hackers, and is able to perform significantly better than humans in terms

1. Funded by European Union Horizon 2020 research and innovation programme under grant agreement No. 700071 for PROTECTIVE project
2. Irish Research Council (IRC) employment-based funding with Ericsson

of interpreting and analysing large data sets. An AI based platform facilitates the building of systems that can detect and act on security threats, either before or immediately after the information has been compromised. The impact of AI on cybersecurity is significant and will continue to grow. Technology giants like Microsoft deploy AI based Intrusion Detection Systems to automate the detection of attacks at data entry points (Horvitz 2022). They aim to answer the key questions about anomalous attack data patterns which remain hidden in the unstructured raw data. Multilayer perceptron models, also known as neural networks, are attracting interest, as these do most of the feature engineering work compared to alternative machine learning methods. Traditional signature based methods such as antivirus fail when exposed to new threats. This has highlighted the importance of anomaly detection and has resulted in drawing the researcher's attention towards adapting autonomous and intelligent systems.

Intrusion Detection Systems (IDS) are a crucial requirement to safeguard an organization's electronic assets. For sequential data the detection of anomalous sequences using Intrusion Detection System can benefit greatly from the use of AI, machine learning and in particular Artificial Neural Networks (ANNs). Recurrent neural networks (RNNs) have proved to be a powerful way of analysing sequential data and have shown impressive results in the area of language modelling (Shen, Huang, Gao & Chen 2017, Cho et al. 2014). Techniques designed for language modelling can be applied to analyzing sequences of logged events such as analyzing system call sequences which are used in Host-based Intrusion Detection Systems. One of the main tasks in language modeling is to predict the next word in the sequence. This corresponds to predicting the next system call in a sequence of system calls. This can be applied to anomaly detection as unexpected, low probability sequences can be identified as anomalies.

1.1.2 Anomaly Detection in Telecommunication

Year on year, an exponential growth in the amount of data has been observed within any organization. It is imperative to perform telecommunications network monitoring and analysis to detect anomalous behaviour on the network. The research conducted in recent years has been to identify potential bottlenecks before they arise in 5G networks. Signature based models (rules derived by network operators) fail if a novel anomalous session occurs. A signature-based approach cannot handle complex data patterns, and labeled data is not always readily available, so an unsupervised approach is needed.

When analyzing traffic fluctuations, deep learning is used to learn representations of data with multiple levels of abstraction in multiple layers of processing. The unsupervised learning of representations in addition to parallel computing implementations such as GPU attempts to perform multiple, simultaneous computations. Using neural network algorithms, researchers can now develop and test real-time network performance management algorithms for the telecommunication industry. The application of an optimized and computationally efficient anomaly detection system can be valuable support for network operators. This would be able to identify the anomalies in Key Performance Indicators (KPIs) over a period of time (more details about KPIs in section 4.1.2.1).

Although DNNs achieve the state-of-the-art performance, this is not enough. They lack the ability to bring intuitiveness to the end user in terms of explaining the output of the model, that is eXplainable AI (Samek et al. 2019). Recently, it is becoming more apparent that eXplainable or interpretability of neural networks plays a major role.

This research will address network performance management based KPI vulnerabilities in the telecommunications industry through novel architectures. The results will therefore provide insights for the network operation engineers. In addition the implementation of deep neural networks with an 'interpretable mechanism' will in effect change what is normally a black box technique into a white box technique enabling location of contributing factors causing anomalies in the network. The research aims to address the broader horizon of discovering unknown events by developing efficient, interpretable and fast neural network-based solutions.

The aim of this project is to investigate state-of-the-art techniques in unsupervised Deep neural networks to perform anomaly detection and extend to provide optimal feedback and interpretation to Engineers and Analysts.

Note: The research has been carried out on Ericsson real network performance management datasets, which is their proprietary information and cannot be released in public domain. All released research results will carefully avoid mention of identifiable customers or identifiable network users. Standard data security precautions are in place in Ericsson's standard operating procedures. While the work has been implemented and evaluated using prototype Ericsson's datasets and testing environments, the results has been demonstrated using anonymous data.

1.2 Research Questions

The main questions in this research are identified as follows:

"*Is it possible to design and develop high performance and interpretable anomaly detection solutions based on artificial neural networks?*"

With the goal of anomaly detection, this thesis addressed two main requirements. Firstly, reducing computational resources. Secondly, explaining the basis of the model predictions. Therefore, this thesis addressed the following research questions.

The first research question is

" *Is it possible to design and develop neural network based anomaly detection solutions with reduced training and testing times, and get optimal anomaly detection results, in the areas of cyber security sequential data ?* "

Cyber security has become one of the most challenging aspects of modern world digital technology and it has become imperative to minimize and if possible avoid the impact of cybercrimes. Host based intrusion detection systems help to protect host from various kinds of malicious cyber attacks. One approach is to determine normal behaviour of a host based on sequences of operation system calls made by processes in the system. The proposed work describes a computational efficient anomaly based Intrusion Detection System based on Recurrent neural networks. Using Gated Recurrent Units rather than the usual LSTM networks, the model achieved state-of-the-art results with good AUC with order of magnitude reduced training times. The incorporation of stacked CNNs with GRUs leads to improved anomaly Intrusion Detection System. The Intrusion Detection System predicts the probability of a particular call sequence based on language models trained on the ADFA-LD data set (Chawla et al. 2018). Sequences with a low probability of occurring are classified as an anomaly.

Another approach is to design a computationally efficient anomaly based Intrusion Detection System using an Encoder-Decoder mechanism. The Bidirectional Encoder and a unidirectional Decoder is trained on normal call sequences and predicts the target sequence to reconstruct the normal behavior. The reconstructed sequences are then used to detect the anomalies.

The second research question is

" *Is it possible to design a framework for Deep neural network Anomaly Detection, with enhanced interpretability and performance, to support telecommunications network troubleshooting and root cause analysis?* "

As telecom networks generate high-dimensional data, it becomes important to support large numbers of coexisting network attributes and to provide an interpretable and eXplainable Artificial Intelligence (XAI) anomaly detection system. Most state-of-the-art techniques tackle the problem of detecting network anomalies with high precision but the models don't provide an interpretable solution. This makes it hard for operators to adopt the given solutions. One of the novelties of this work is the proposed Cluster Characterized Autoencoder (CCA) architecture which improves model interpretability by designing an end-to-end data driven AI-based framework. Candidate anomalies identified using the feature optimised Autoencoder and entropy based feature ranking are clustered in reconstruction error space using subspace clustering. The aim of CCA is to implement the interpretable framework using reduction and categorization of anomalies to help the analyst to find the most relevant anomalies faster and to aid in faster root cause analysis. Therefore, the proposed solution provides better support for the network domain analysts with an interpretable and explainable Artificial Intelligence (AI) anomaly detection system. Work from this reference architecture has been applied to and tested on a set of network cell trace data.

1.3 Contributions & Publications

The idea and the contributions presented in this thesis are part of several peer reviewed research papers. In this subsection, the list of publications emanating verbatim from this thesis is given. Figure 1.1 presents a hierarchical view of the proposed anomaly detection algorithms. As shown, the anomaly detection algorithms have been evaluated in two domain, cyber security and telecommunication.

1.3.1 Major Contribution & Associated Papers

In this thesis, an Interpretability Framework for Network Anomaly Detection is presented which forms the basis of the novel Cluster Characterized Autoencoder (CCA) architecture to provide interpretable feedback to the user. The implementation of subspace clustering processes enables the analyst to find the most relevant anomalies faster using reduction and categorization of candidate anomalies. The reconstruction error of features using a Kernel SHapley Additive exPlanations (SHAP) explainer aids in faster root cause analysis.

1. Published: **Towards Interpretable Anomaly Detection: Unsupervised Deep neural network Approach using Feedback Loop.** (section 4.2)
   Chawla A, Jacob P, Farrell P, Aumayr E and Fallon S, Towards Interpretable Anomaly Detection: Unsupervised Deep Neural Network Approach using Feedback Loop, NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1-9, doi:10.1109/NOMS54207.2022.9789914.

2. Published: **Interpretable Unsupervised Anomaly Detection for RAN Cell Trace Analysis** (section 4.1)
   Chawla A, Jacob P, Feghhi S, Rughwani D, van der Meer S, Fallon S. Interpretable Unsupervised Anomaly Detection for RAN Cell Trace Analysis. In 2020 16th International Conference on Network and Service Management (CNSM) 2020 Nov 2 (pp. 1-5). IEEE.

1.3.2 Minor Contributions & Associated Papers

The incorporation of stacked Convolutional Neural Network (CNNs) with Gated Recurrent Units (GRUs) leads to improved anomaly Intrusion Detection System with reduced training times. The concept leverages the conditional probability prediction with trained neural network model to classify sequences as anomalies. Further, Bidirectional based autoencoders improved the Intrusion Detection System AUC. Lastly, performing real time anomaly detection for IoT data using Conv-LSTM autoencoder model as a cloud native application.

3. Published: **Host based intrusion detection system with combined CNN/RNN model** (section 3.1)

Chawla A, Lee B, Fallon S, Jacob P. Host based intrusion detection system with combined CNN/RNN model. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases 2018 Sep 10 (pp. 149-158). Lecture Notes in Computer Science including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics. Springer, Cham. ISBN 978-3-030-13453-2.

4. Published: **Bidirectional LSTM autoencoder for sequence-based anomaly detection in cyber security.** (section 3.2)

   Chawla A, Jacob P, Lee B, Fallon S. Bidirectional LSTM autoencoder for sequence-based anomaly detection in cyber security. International Journal of Simulation–Systems, Science & Technology. 2019.

1.3.3 Publications

Here Fig. 1.1 includes references to the five publications stated above.

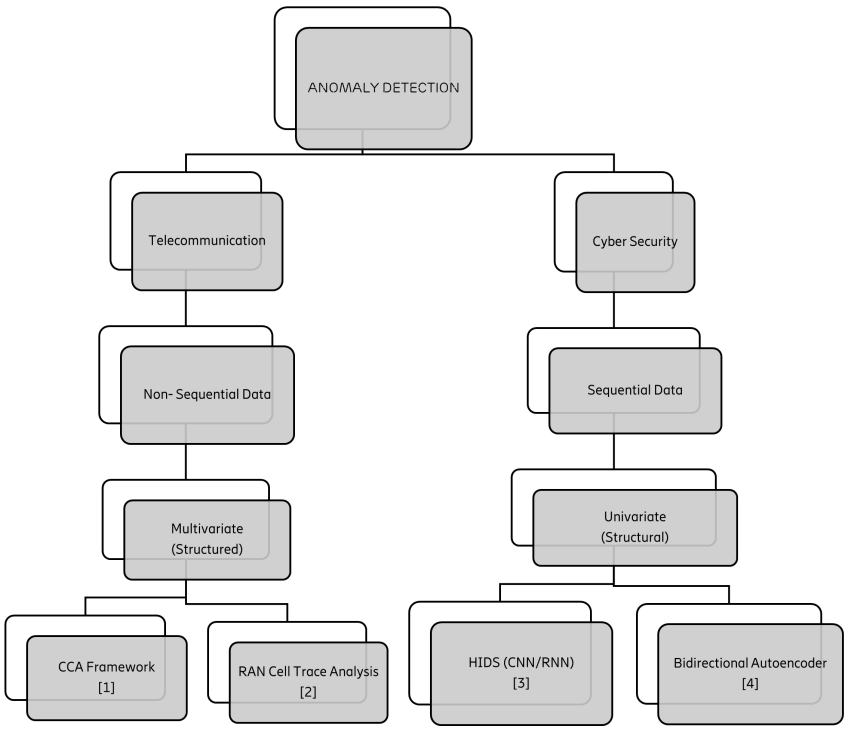


Figure 1.1: Hierarchical structure of Anomaly Detection Algorithms & Contributions

1.3.4 Patent

This patent was filed by Ericsson based on the work been conducted as part of the thesis.

Patent application: PCT/EP2022/055178

Filing date: 1 March 2022

Title: Network Node, and method performed therein for Anomaly Detection

1.4 Thesis outline

Chapter 2 outlines the literature review, the concepts, definitions, algorithms, and applications for Neural Networks, followed by introduction to anomaly detection in Cyber Security and Telecommunications. As the focus turns to interpretable anomaly detection solutions (eXplainable Artificial Intelligence), this chapter also presents the methodology of proposed different model agnostic methods, while distinguishing explainability and interpretability terminology. Followed by anomaly detection performance metrics.

Chapter 3 outlines the design and evaluate the anomaly detection algorithms based on Host based Intrusion Detection System using CNN/RNN based models and bidirectional based LSTM models. The computationally efficient models designed significantly improved the performance in terms of training and testing times with higher accuracy when compared to state-of-the-art. The first section introduces a prediction-based approach to anomaly detection using sequential data (ADFA-LD). Using autoencoders as a basis, the second section designs anomaly detection based on reconstruction errors for the ADFA-LD data set.

Chapter 4 outlines the main contributions of the PhD thesis work, the interpretability and the Cluster Characterized Autoencoder architecture (CCA) which improves the model performance by designing an end-to-end data driven AI-based framework. The first section discuss the approach applied on a network cell trace data set with global and local interpretations of the results. An extension on the first approach, the second section considers an interpretable anomaly detection solution that makes use of subspace clustering and an internal feedback loop.

Finally, chapter 5 contains conclusions and analysis of the research. Proposals for future work are also included.

Chapter 2

Literature Review

This section provides a review of the state-of-the-art in the areas that are relevant to the research objectives. Section 2.1 reviews state-of-the-art Deep Neural Networks techniques. Various activation functions been used for the purposes of experiment and evaluation are introduced in section 2.1.2. This is followed by feed-forward Neural Networks, Deep Neural Networks, Batch Normalization, and different variants of Recurrent Neural Networks. Convolutional Neural Networks and autoencoders are introduced in section 2.1.6 and 2.1.7 respectively. Section 2.2 reviews HIDS and NIDS based Anomaly Detection methodologies, inspired by Natural Language Processing. Section 2.3 reviews the need for machine learning in the field of telecommunications. Section 2.4 reviews Interpretable Artificial Intelligence, followed by Post-hoc Interpretation, visual analysis, SHAP and kernel explainer for Autoencoders. Section 2.5 discusses the subspace clustering technqiue. Finally, the discussion on the performance metrics in section 2.6.

2.1 Neural Networks

The term “neural network” is an information processing paradigm in reference to neurobiology which works on the concept of a brain to facilitate the model functioning (García-López et al. 2010). They are a set of algorithms designed to recognize and memorize the hidden patterns in the underlying raw data, similar to the way human brain works. The components of a neural network are listed in Table 2.1.

In general, as shown in Fig. 2.1, an ANN comprises of a set of artificial neurons arranged in layers. It is a powerful tool for solving problems such as classification or prediction. Feed-forward neural networks are the most commonly encountered and are used in many diverse applications. Backpropagation algorithm is the mostly used method in the training of feed-

Table 2.1: Components of Neural Network

Input	Raw input supplied to the neural network
Hidden Layers	Collection of neuron units to process the information
Activation Function	Defines output of a node
Loss function	Mechanism to evaluate a model algorithm
Optimizer	Determines the learning rate of an algorithm
Output	Data after being processed by the model.

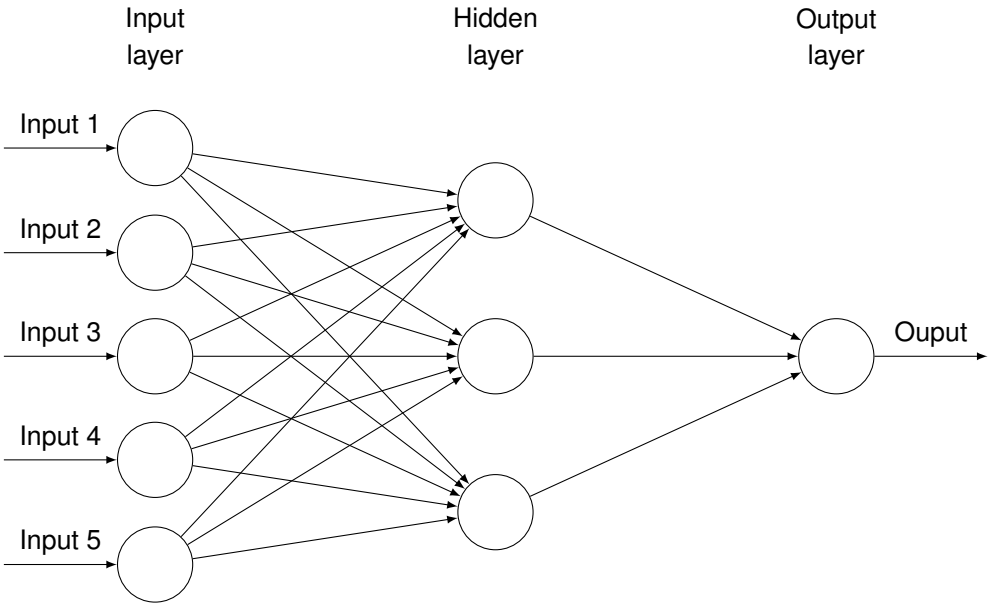


Figure 2.1: Artificial Neural Network

forward neural networks. Backpropagation in neural network is a short form for “backward propagation of errors”. The Backpropagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time.

2.1.1 Activation Function

The main purpose of the activation function is to convert an input signal of a node in an ANN to an output signal. This signal becomes an input to the nodes at the next layer (Jain et al. 1996). In an ANN layer, the nodes computes an output by processing a linear combination of weighted real-valued inputs through a nonlinear activation function. The output signal would be a simple linear function if the activation function was not applied. A linear equation is easy to solve, but its complexity is limited and it has less power to learn complex functional mappings from input data. An ANN without an activation function would simply be a linear regression model with limited appeal. Therefore, an ANN would not be able to learn effectively without an activation function (Specht 1991).

2.1.1.1 Sigmoid

The Sigmoid function has the mathematical form:

$$S(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

The sigmoid function domain is R and its range is [0,1]. Large negative numbers are effectively 0 and large positive numbers effectively 1 (Park et al. 1991). The sigmoid function (Fig. 2.2) has historically been used because it mimics well the firing rate of a neuron; from not firing at all (0) to fully saturated firing at a maximum frequency.

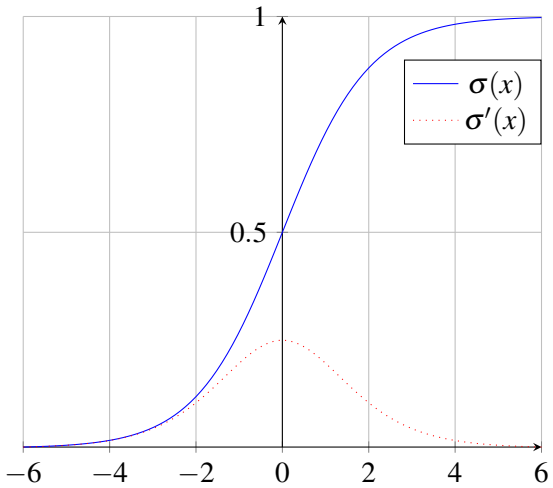


Figure 2.2: Sigmoid Activation Function

2.1.1.2 Tanh

Like the sigmoid function, tanh activations saturate, but its output is zero-centered as shown in Fig. 2.3. Note also that the tanh neuron is just a scaled sigmoid neuron. Its mathematical formula is:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.2}$$

The output ranges from -1 to 1, which makes each layer's output more or less centered around 0 at the beginning of the training. This often results in faster convergence.

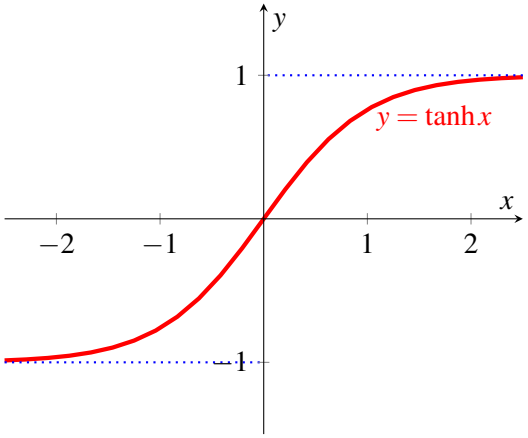


Figure 2.3: Tanh Activation Function

2.1.1.3 Relu

Rectified Linear Unit (Relu) as shown in Fig. 2.4 is a non-linear activation function used in neural networks. There are several advantages and disadvantages with the use of ReLUs (Nair & Hinton 2010). The main advantages is the convergence of stochastic gradient descent, compared to the sigmoid and tanh functions, is greatly accelerated. It is argued that this is because of its linear, unsaturated shape. In addition, tanh and sigmoid neurons involve costly operations (exponentials, etc.), whereas the ReLU can be implemented simply by setting a zero activation matrix.

$$f(x) = max(0, x) \tag{2.3}$$

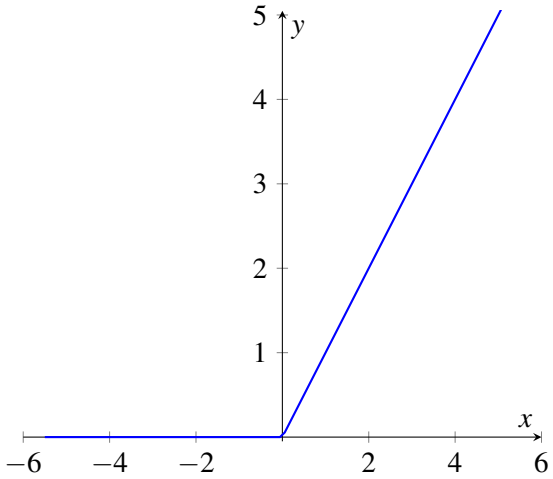


Figure 2.4: Relu Activation Function

However, ReLU units can be fragile and die during training. For example, a large gradient that flows through a ReLU neuron can update the weights so that the neuron never again activates at any data point. If this happens, the gradient that flows through the unit from that point is zero forever. The ReLU units can therefore die irreversibly during training.

2.1.2 Feed-forward Neural networks - Perceptron Model

A feedforward neural network is one of the most basic and simplest Artificial Neural Network (ANN). For instance, Fig. 2.5 consists of a neural network with input as X_1, X_2, X_3. The specification of what a layer does to its input data is stored in the layer's weights, which in essence are a bunch of numbers. In technical terms, the transformation implemented by a layer is parameterized by its weights (also known as the parameters of a layer). These weights encode the information learned by the network from exposure to training data. In addition bias vector (typically set to 1.0) is applied to the model to help learn patterns and allows the output of an activation function to shift to the left or right. The activation function such as Sigmoid, Tanh or relu applied to the non- linear dataset helps in predicting the outcome of a model. The output of a neural network is defined as:

$$y = activation(dot(W, input) + b) \tag{2.4}$$

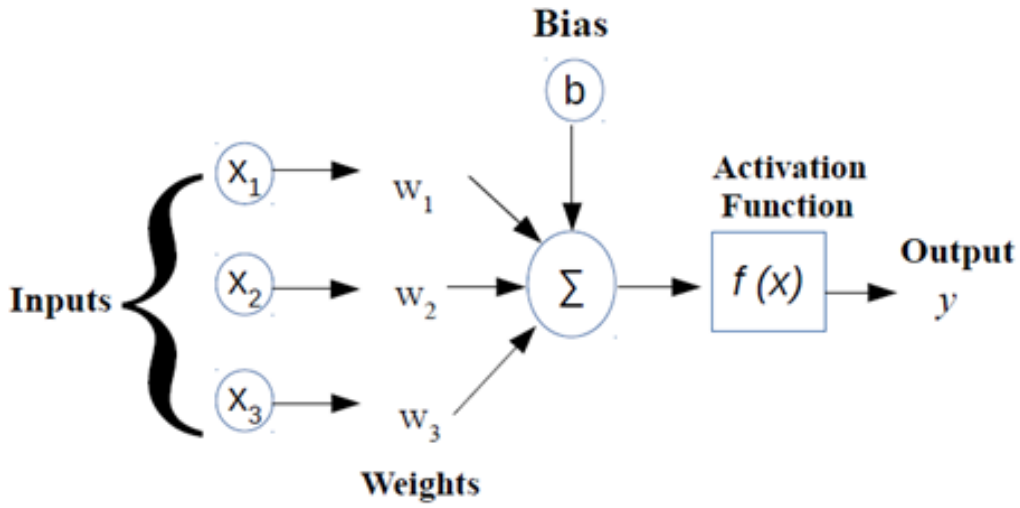


Figure 2.5: Artificial Neural Network - Perceptron Model

Since the weights were randomly initialized, the delta of error between predicted and actual output is very high with less accuracy and high loss score. The weight values are adjusted and trained by the Backpropagation algorithm as shown in Fig. 2.6. During back-propagation, the neuron weights are adjusted to produce a smaller error value (Benvenuto & Piazza 1992). As the back propagation algorithm with a fixed learning rate shows low efficiency, Yu et al. (1995) introduced the concept of dynamic learning rate optimization of the backpropogation algorithm. This process defined how an optimum learning rate accelerated the convergence of the backpropogation algorithm producing a remarkable reduction.

2.1.3 Deep Neural Networks

Deep Neural Networks (DNNs), in the field of computer science, have revolutionized many sectors of the economy ranging from Google translate to driverless cars to personal cognitive assistants (Sejnowski 2018). Technically, an ANN that has a lot of layers is a DNN. They achieved state-of-art results in many research areas such as autonomous vehicles (Kisačanin 2017), medical image processing (Shen, Wu & Suk 2017), handwritten character recognition (Vaidya et al. 2018), and Natural Language Processing (Kamath et al. 2019). DNNs are more complex to train than shallow neural network as they involves tens and in some cases up to thousands of successive layers of representation.

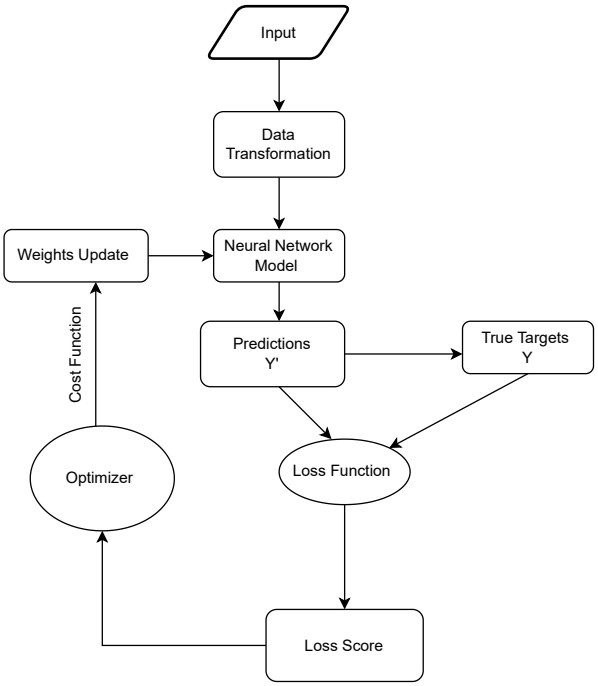


Figure 2.6: Optimizing Neural Network

Deep neural networks leverage complex compositions of linear/non-linear functions to learn expressive representations. The feature representations extracted by deep learning models preserve the discriminative information that helps separate abnormal instances from normal instances.

Like processing of scientific algorithms, the extensive computations required to train and use deep neural networks naturally lend themselves to parallel computing implementations and GPU technology is ideally suited for parallel computing tasks. The author mentioned that GPUs differ from CPUs where GPUs are optimized for throughput instead of latency, as throughput is more important for massive computations (Caulfield 2009). In general, large operations like matrix multiplication, or element-wise operations with large inputs, will be significantly faster. Taking advantage of existing GPU hardware for computationally intensive deep learning implementations will generate performance acceleration.

As a result, much current research in this area focuses on GPU accelerated deep learning. There are already some machine learning toolkits utilizing GPU direct programming, mainly for neural networks and Support Vector Machine (SVM). For example, the Deeplearning4j project actually integrates with the CUDA kernels to perform operations in the GPUs' un-

derlying storage (*Deeplearning4j* 2022). CUDA is a software layer providing APIs that gives direct access to the GPU's virtual instruction set and parallel computational element. Another tool, apart from Deeplearning4j, that allows developers to directly introduce deep learning algorithms into their systems is the R package Deepnet (Rong 2009).

2.1.4 Batch Normalization

Though using activation functions, such as ReLU helps in reducing the danger of vanishing/exploding gradients, but they do not guarantee they won't appear again during training of the model. In order to handle the problem of gradient propagation, Ioffe & Szegedy (2015) proposed a technique called as Batch Normalization (BN).

Normalization is a broad category of methods that seek to make different samples seen by a machine-learning model more similar to each other, which helps the model learn and generalize well to new data. The technique comprises of adding an operation in the model before or after the activation function of each hidden layer. The most common form of data normalization is by centering the data on 0 by subtracting the mean from the data, and giving the data a unit standard deviation by dividing the data by its standard deviation. In effect, this makes the assumption that the data follows a normal (or Gaussian) distribution and makes sure this distribution is centered and scaled to unit variance.

In order to zero-center and normalize the inputs, the algorithms needs to estimate each input's mean and standard deviation. It does by evaluating the mean and standard deviation of the input over the current mini-batch (subset of data during one iteration) of data seen during training. The main effect of batch normalization is that it helps with gradient propagation, much like residual connections and thus allows for deeper networks. Some very deep networks can only be trained if they include multiple BatchNormalization layers. The BN operation algorithm is summarized as:

$$\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} x^{(i)} \tag{2.5}$$

$$\sigma_B{}^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} (x^{(i)} - \mu_B)^2 \tag{2.6}$$

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{{\sigma_B}^2 + \varepsilon}} \tag{2.7}$$

$$z^i = \gamma \otimes \hat{x}^{(i)} + \beta \tag{2.8}$$

From the above equations,

- μ_B is the vector of the input calculated over the mini-batch *B*.
- σ_B is the standard deviation of the input over mini-batch.
- m_B is the number of instances in the mini-batch.
- $\hat{x}^{(i)}$ is the vector of zero-centered and normalized inputs for instance i.
- γ is the output scale parameter.
- $\otimes$ represents the cross multiplication of the element by it's corresponding output scale parameter.
- β is the output shift (offset) parameter vector for the layer (it contains one offset parameter per input). Each input is offset by its corresponding shift parameter.
- ε is to avoid divide by zero, also known as smoothing term in neural networks.
- z^i is the output of the batch normalized operation. It is a re-scaled and shifted version of the inputs.

The BN estimates the above statistics during the training by using a exponential moving average of the layer's input means and standard deviation. Ioffe & Szegedy (2015) demonstrated that this technique acts like a regularization, considerably improved the overall training performance of the DNNs when experimented with ImageNet classification task. By using higher learning rates, they were able to achieve the same accuracy with 14 times fewer training steps and significantly speeding up the learning process.

2.1.5 Recurrent Neural Network

Recurrent Neural Network (RNN) as shown in Fig 2.7 are another form of neural network well known for natural language processing tasks (NLP), such as (Gers et al. 2000, Shi et al. 2019). Fig 2.7 shows a basic block of RNN, in which the output of the current input depends on the past information.

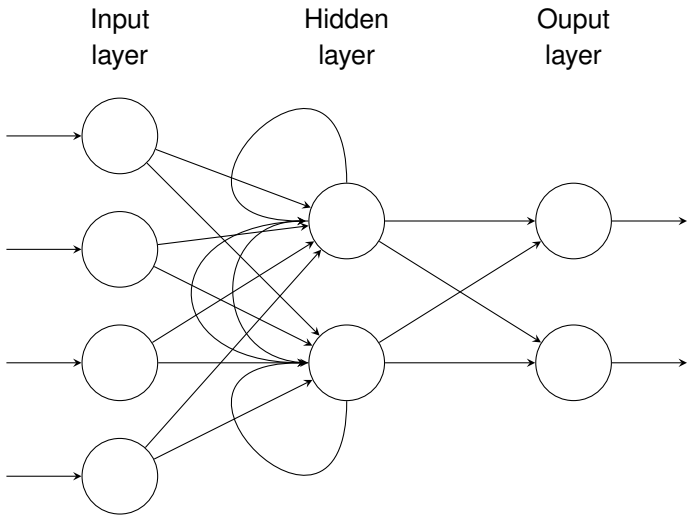


Figure 2.7: Basic block of RNN

RNNs have been shown to be able to process sequential data much faster than conventional neural networks. They consist of hidden states that feeds back itself recurrently as shown in Fig 2.7. The hidden state value is determined by the current timestep input and the value from the previous timestep. RNNs leverage the sequence learning capability, where the output at each timestep is determined by the information from the past inputs, trained using Backpropagation Through Time (BPTT) (Werbos 1990). BPTT is the application of the Backpropogation algorithm which is applied to the sequential data, for instance time series data.

The hidden state of RNN is defined as:

$$h_t = f_w(h_{t-1}, x_t) \tag{2.9}$$

where h_t denotes the new hidden state at current timestamp 't', h_{t-1} denotes the old hidden state value from previous timestamp "t-1", f_w represents some non-linear activation function with parameters W and x denotes the input vector at timestep "t".

The unrolled state of a RNN at a timestep 't' can be seen in Fig 2.8. As shown on the left, RNNs can be unfolded to become a regular neural network. In this diagram, a single node (on the left) represents a complete layer in the RNN. The different time steps are visualized and information is passed from one time step to the next. Backpropagation applied to this unfolded network is known as Backpropagation Through Time and can be used to train the RNN. In BPTT, the conceptualization of unrolling is required since the error of a given timestep depends on the previous time step. Within BPTT, the error is backpropagated from the last to the first timestep, while unrolling all the timesteps. This allows calculating the error for each timestep, which allows updating the weights.

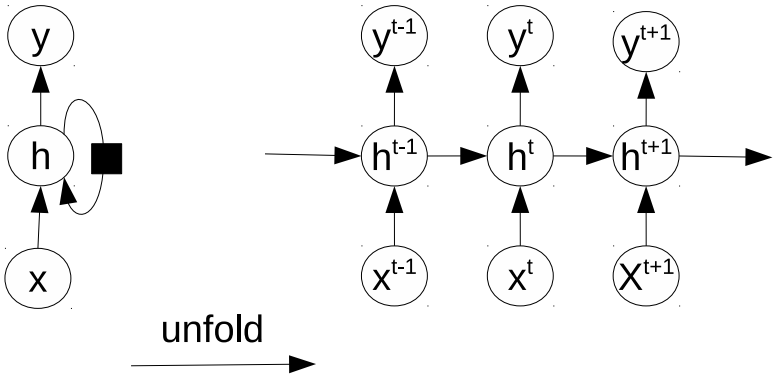


Figure 2.8: Unfolding an RNN

Note: The parameters and functions are shared at every timestep. In this thesis, RNNs form the basis for the approach to evaluate Host based Intrusion Detection System.

2.1.5.1 Limitations of RNN

There are two major limitations that occur during the backpropagation of RNNs. Firstly, the problem of "exploding gradients". This happens when the gradients is too large and their value becomes a NaN due to the unstable process. Secondly, while RNN can be trained to capture short term dependencies between time steps, it has proved difficult to train RNNs to capture long term dependencies due to "vanishing gradient" problem (Grosse 2017).

To overcome this, special types of RNNs have been designed,in particular Long Short Term Memory networks (LSTM) and Gated Recurrent Units (GRU). Recurrent Neural Networks work effectively in case of small range dependencies, but often proves to be inefficient in long range term dependencies. With longer sequences, arises the problem of Back Propagation Through Time mechanism (BPTT) and hence result in vanishing gradient descent problems.

2.1.5.2 Long Short Term Memory (LSTM)

Hochreiter & Schmidhuber (1997) proposed a novel recurrent architecture which was capable of retaining the "important" information over long sequences, known as Long Short Term Memory neural network (LSTM). LSTM networks are able to learn and store information over extended time intervals by memorizing certain patterns. They are a type of recurrent neural network capable of learning order dependence in sequence prediction problems, mainly required for NLP tasks such as sequence language modeling. LSTMs are trained via recurrent Backpropagation Through Time (BPTT).

LSTM consists of "context" or "cell" which works like a conveyor belt and runs over the sequence of input, hence stores additional state value over time. It replaces traditional RNN with input gates, output gates and forget gates. Each gate provide access to the cells in such a way that values can be stored in the cell for either short or long periods of time and removed when no longer needed.

- **Forget gate:** Gers et al. (2000) proposed the concept of forget gate which is responsible for an LSTM cell to reset itself at appropriate times, as shown in Fig. 2.9. It scales the internal state of the cell and optimizes the performance of LSTM using the following function:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.10}$$

  The forget gate, also known as the remember vector, decides what information needs to be retained and what needs to be discarded. The information from cell state is passed through the first sigmoid activation function where "0" means to forget and "1" means to retain. This forms the first step in the process. The outputted values are then sent up

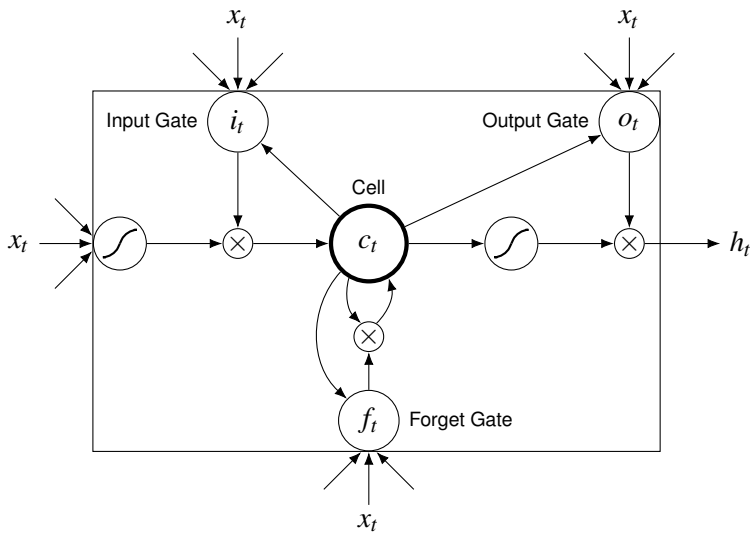


Figure 2.9: LSTM cell Architecture

and pointwise multiplied with the previous cell state. This pointwise multiplication means that components of the cell state which have been deemed irrelevant by the forget gate network will be multiplied by a number close to 0 and thus will have less influence on the following steps.

- **Input gate:** The next step involves the input gate, which controls the flow of new input information into the memory cell. This gate determines which new information would be added to the cell state, given the previous hidden state and new input. The activation function sigmoid regulates the process of adding or removing information by the following function:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.11}$$

Here W_i is the weight matrix applied to the input activations, b_i is the bias vector, σ is the sigmoid function. The cell carries the information throughout processing of the input sequences. It represents the learnings across all of the time.

$$c_t^{\sim} = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.12}$$

Where, cell and hidden state is defined as:

$$c_t = f_t \bigodot c_{t-1} + i_t \bigodot c_t^{\sim} \tag{2.13}$$

$$h_t = o_t \bigodot tanh(c_t) \tag{2.14}$$

Where, $\bigodot$ represents element wise operation on vectors, is the cell state memory at timestamp (t) and represents new candidate values using tanh layer.

- **Output gate:** The output gate controls the job of selecting useful information from the current cell. It outputs the flow of cell activations into the rest of the network by the following function:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.15}$$

The output gate decides the new hidden state. To decide this, it depends on the newly updated cell state, the previous hidden state and the new input data.

2.1.5.3 Sequence to Sequence Learning

Sutskever et al. (2014) evaluated Sequence to Sequence (Seq2Seq) learning using Deep Neural Networks (DNN) model with LSTM units. Seq2Seq is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French), derived from the concept of NLP tasks as shown in Fig. 2.10. The multi-layered LSTM model performed remarkably well on English to French translation task. The authors outlined a comprehensive review of the state-of-the-art in sequence to sequence learning, which typically involves carrying information from previous time step to later ones.

This work outlined the methodology implemented for mapping the input sequence to a fixed sized dimensionality vector. Further the target sequences were decoded using the vector information. The authors concluded that LSTM based approach worked significantly well on many other sequence problems obtained by reversing the words in a sentence.

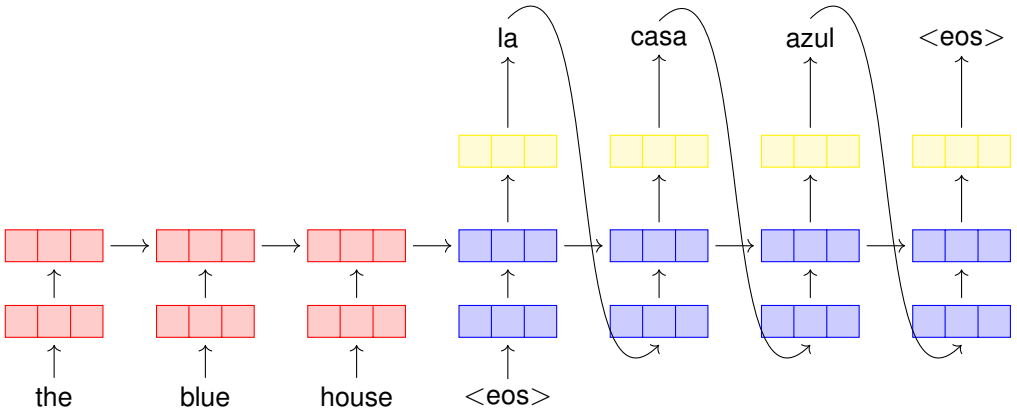


Figure 2.10: Illustrating Sequence to Sequence Learning

2.1.5.4 Gated Recurrent Unit (GRU)

Recently another type of recurrent unit, known as GRU was proposed by Hochreiter (1998) to solve the problem of vanishing gradient. The functionality of GRU is almost similar to LSTM, but with the difference of no separate memory cells. GRU's architecture are internally simpler which consists of an update gate and a reset gate.

- **Update Gate:** The update gate (shown as z in Fig 2.11) is responsible for how much of previous information to keep around. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$.

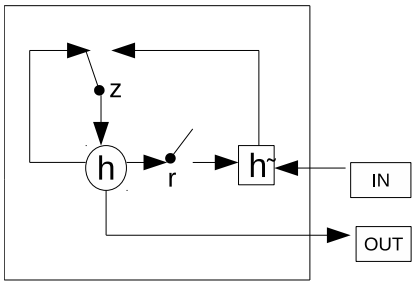


Figure 2.11: Gated Recurrent Unit

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.16}$$

Where σ is non-linear activation function, x_t and $h_(t-1)$ denotes the input and the hidden states at time "t" and "t-1" respectively. W_z and U_z represents the weight matrices been learned.

- **Reset Gate:** This reset gate defines how new input is combined with the previous value. Essentially, this gate is used from the model to decide how much of the past information to forget.

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{2.17}$$

The candidate activation $\tilde{h}$ is computed as:

$$\tilde{h} = tanh(W x_t + U(r_t \bigodot h_{t-1})) \tag{2.18}$$

where r_t is reset gate and $\bigodot$ is an element-wise multiplication.

Fu et al. (2017) stated that experimentally GRU models converges faster when compared to an LSTM neural network model. Chawla et al. (2018) have shown that CNN/GRU combination improved the performance for evaluating a call sequence data. The model achieved comparable state-of-the-art results with substantial reduced training times.

2.1.5.5 Bidirectional Recurrent Neural Networks

With respect to natural language processing tasks, such as speech recognition work by Sutskever et al. (2014), bidirectional recurrent neural networks based models have resulted in achieving the state-of-the-art performance (Birnie & Hansteen 2022). Bidirectional recurrent neural networks shown in Fig 2.12 seek information from both earlier and later (chronologically and anti chronologically) in the sequence. The output of the forward and backward layer is combined at each time step by one of the following merge modes means:

- Concat: The outputs are concatenated together, hence providing double the number of outputs to the next layer.
- Mul: The outputs are multiplied.
- Sum: The outputs are added.
- Ave: Average of the two output is taken.

The neural network is principally trained using an algorithm such as Backpropagation Through Time (BPTT) which solves the vanishing/exploding gradient problem. The final prediction of the neural network at time (t) is calculated using the following equation:

$$y^{<t>} = g(W_y[a^{\rightarrow <t>}, a^{\leftarrow <t>})] + b_y) \tag{2.19}$$

Where, g denotes the non-linear activation function, W_y denotes weight matrix, $a^{\rightarrow <t>}$ $a^{\leftarrow <t>}$ denotes forward and backward recurrent components respectively. b_y denotes bias and $y^{\wedge <t>}$ denotes output at timestep "t". Here figure 2.12 shows that predicted output at $y^{\wedge <1>}$, $y^{\wedge <2>}$, $y^{\wedge <3>}$ is calculated for each of the recurrent unit inputs defined, for forward and backward connected to each other.

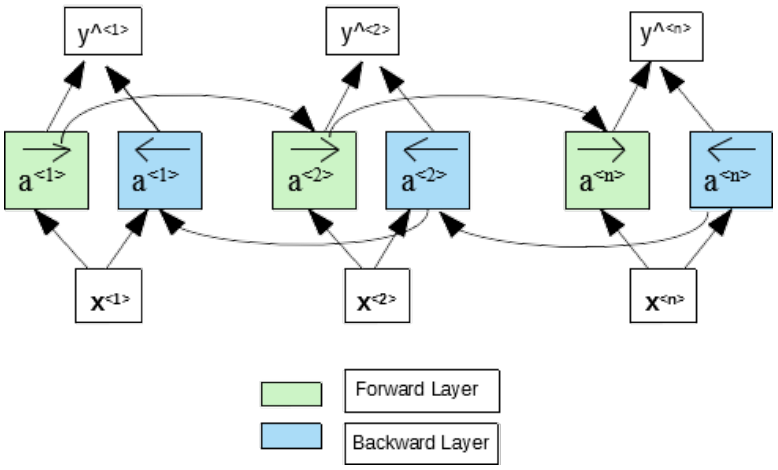


Figure 2.12: Bidirectional Recurrent Neural Network

Note: BRNNs require input of the entire sequence of data before making any predictions. Also, when implementing an autoencoder using BRNNs there is no need to reverse the source/target sequence, as BRNNs takes information from both the forward and backward direction.

2.1.6 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of a neural network which emerged primarily in the domain of image processing since 1908s. LeCun et al. (1998) proposed LeNet-5 architecture, which achieved the milestone to recognize the handwritten check numbers by creating the feature maps that highlight the most relevant parts of the input image.

2.1.6.1 The CNN Architecture

In the case of 2D data, convolution is effected by a 2D filter sliding over the image and applying some function to the covered part of the image to produce the output. By using suitable functions, patterns in the image can be detected, for example, taking the difference between pixels can be used to detect edges. The following are the key components of CNN:

- **Convolutional Layers**: Convolutional Layers form the most important building block of a CNN. The neurons in the first convolutional layers are not connected to every single pixel in the input. Instead, they are connected to pixels in their respective fields. This type of architecture allows to concentrate on the specific features (as shown in Fig 2.13) in the hidden layers. The objective of the convolutional layer is to extract low-level features.

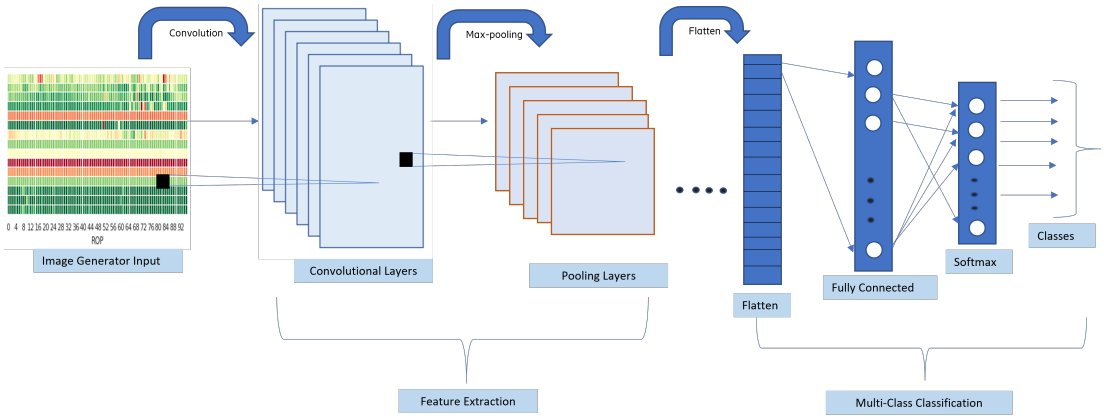


Figure 2.13: An Example of CNN Blocks

- **Kernel**: CNNs contains a series of filters known as convolutional kernels. A CNN kernel consists of a feature map matrix that moves over the input data and performs the dot product with the sub region of the input data. Example, CNN kernels sharpens a picture by sharpening the edges with regards to sharpening.

The edge detection kernel features an edge-detecting kernel with a center value of 5 when sharpening images. A 3×3 version adds contrast to edges by enhancing their edge quality. In this manner, bright or dark areas are highlighted in a way that makes the image more engaging.

- **Pooling**: A pooling layer reduces the the complexity of the model and input image in order to reduce the computational load, the memory usage and the number of parameters to limit the risk of overfitting. As shown in the Fig. 2.13, each neuron in the pooling layer is connected to the outputs of a limited number of neurons from the previous layer, located within a small rectangular receptive field.

In Fig 2.13, the input generator when passed through the convolutional layers, perform pooling operation and further flattens the data. Flattening in CNN is to convert data into 1-dimensional array to create a feature vector array as an input to fully-connected image classifier model. The final activation function, softmax calculates the probability distribution and classifies the images into different classes.

The softmax score for an instance x, computes a score $s_k(x)$ each class is defined as:

$$s_k(x) = x^T \theta^k \tag{2.20}$$

where each class has it's own dedicated parameter vector θ^k

After computing the score for every class for the instance x, the probability $\hat{p}_k$ that the instance belongs to class k is computed through softmax function defined as:

$$\hat{p}_k = \sigma(s(x))_k = \frac{exp(s_k x)}{\sum_{j=1}^{k} exp(s_j x)} \tag{2.21}$$

where k is the number of classes, s(x) is a vector containing the scores of each class for the instance k. $\sigma(s(x))_k$ is the estimated probability that the instance x belongs to the class k, given the scores of each class for that instance.

2.1.6.2 1-Dimensional CNN for Text

In the case of 1D data, filters slide over sequences extracting a feature map for local subsequences in the data. They create representations for fixed size contexts and the effective context size can easily be made larger by stacking several CNN layers on top of each other. This allows to precisely control the maximum length of dependencies to be modeled. As convolutions are a common operation in computer graphics with direct hardware support on GPUs, CNNs are a more efficient way of extracting local patterns from sequences than RNNs. Note that, pooling is not applied after the convolution operation (Shi et al. 2019). The output from the stacked CNN layers are passed to the RNN which can be used to capture long-range dependencies.

2.1.7 Autoencoders

An autoencoder is a type of neural network that is trained to learn a compressed representation of the input data and to reconstruct the input from this representation (Baldi 2012). Anomaly detection using autoencoders is carried out by identifying the instances which can not be accurately reconstructed, and these are then identified as anomalous instances.

In effect, autoencoders build a model to identify hidden structures and useful features from unlabelled input data. They work on the concept of learning interesting hidden layer representations by limiting the number of neurons in the hidden layer. An Autoencoder shown in Fig 2.14 consist of an encoder, a latent space representation and a decoder. The encoder learns a compressed vector representation of the input data, and the decoder uses this information to reconstruct the input from its hidden representation.

The Encoder-Decoder learns to reconstruct normal behavior, and thereafter the reconstruction error is used to detect anomalies. The model jointly trains the encoder and decoder and applies backpropagation to minimize the reconstruction error (Hecht-Nielsen 1992). The Mean Squared Error (MSE) is used as the loss function for training the model and depends on the the difference between the actual (x) and reconstructed (x') attribute values. The Root Mean Squared Error (RMSE) is used as an evaluation metric to calculate the squared error difference between the observed (x) and predicted (x') values of a set of n (Chai & Draxler 2014).

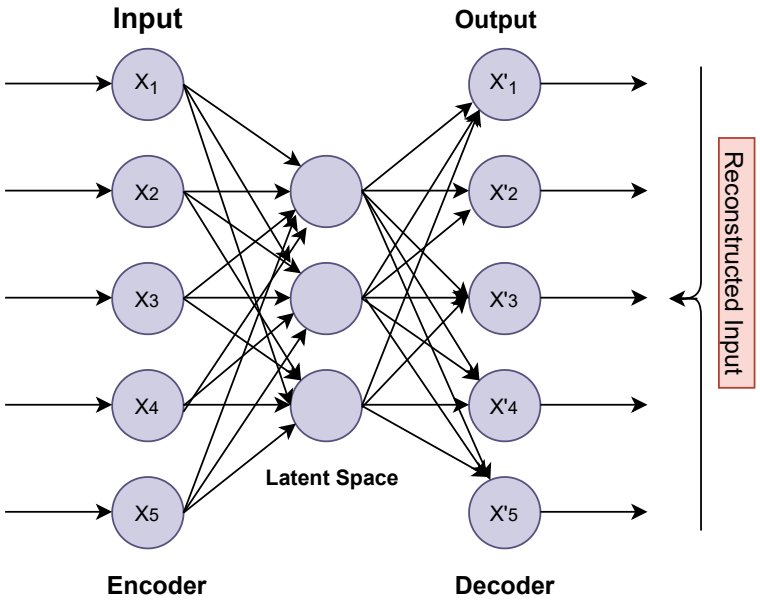


Figure 2.14: Autoencoder

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(x_i - x_i^{'})^2} \tag{2.22}$$

Inspired by work in Malhotra et al. (2015), Zhang et al. (2021) proposed CNN and LSTM based Encoder-Decoder for Anomaly Detection in Multivariate Time Series. The framework called as CNN-LSTMED (Convolutional Neural Networks Long Short-Term Encoder-Decoder). The convolutional neural network encodes the time series data to obtain the encoded sequence, and use the features extracted from the sequence as the input of the nonlinear model LSTM to decode and output the decoded sequence. The Encoder-Decoder was successful in learning to reconstruct the normal behavior, and thereafter the reconstruction error is calculated and the threshold is set to determine the abnormal point.

2.2 Anomaly Detection in Cyber Security

Anomalies can be defined as an unseen pattern that does not match known monitored patterns. An anomaly detection (outlier detection) algorithm is an algorithm which is trained with normal data to abstract the normal behaviour in data as a baseline to identify the anomalies.

*"Outliers are also referred to as abnormalities, discordants, deviants, or **anomalies** in the data mining and statistics literature" (Aggarwal 2017).*

A comparison of signature-based and anomaly detection based approaches is presented in Fig. 2.15. The signature based approach operates in much the same way as a virus scanner by searching for identities or signatures of known intrusion events. Whereas, anomaly detection approach looks for an unseen pattern by building a model of normal behaviour.

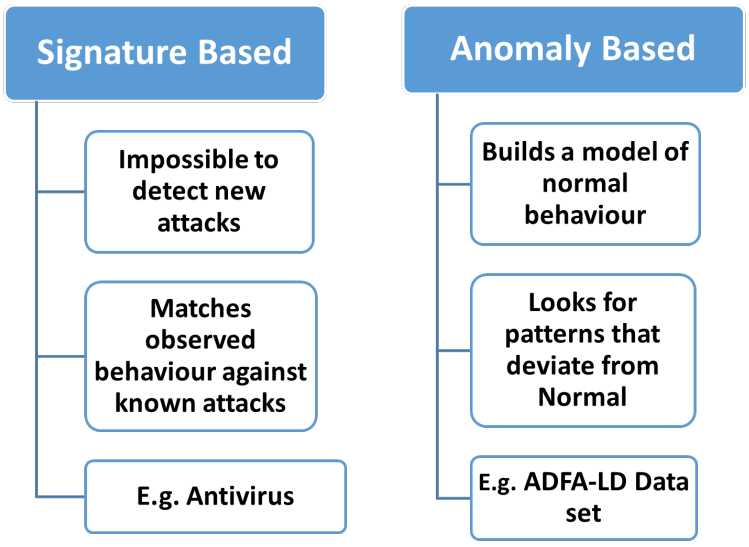


Figure 2.15: Signature Based Methods vs Anomaly Detection

2.2.1 Intrusion Detection System

Anomaly detection is a step in data mining that identifies data points, events, and/or observations that diverge from the pattern of normal behavior. Examples include credit card fraud detection, monitoring machines in a data centre, military surveillance, health sector, image processing etc. An Intrusion Detection System (IDS) is a software application used to scan a network for harmful activity or policy breaches by scanning for traffic for suspicious activity. IDS forms a crucial requirement to safeguard an organization's electronic assets.

There are two types of intrusion detection systems commonly known as Host based Intrusion Detection systems (HIDS) and Network based Intrusion Detection systems (NIDS). Network based intrusion detection systems monitor and analyse network traffic in order to protect a system from network-based threats. A network based Intrusion Detection System collects information from network packets with the aim of detecting the malicious activity in network traffic. A Host based intrusion detection system (HIDS) aims to collect information about events or operating system calls/logs on a particular system.

System calls or kernel calls provide an essential interface between a process and the operating system, and can be used for building a good HIDS. Forrest et al. (1996) suggested that sequences of system calls could be used to capture normal behaviour in a computer system. They proposed a method for defining self for privileged Unix processes, in terms of normal patterns of short sequences of system calls.

Early in 1998 and 2004, Knowledge Discovery in Databases (KDD) and UNM datasets were released for evaluating intrusion detection systems. Table 2.2 shows the distribution of normal and attack records in Network Security Laboratory (NSL-KDD) dataset, a refined version of its predecessor KDD99 data set. Dhanabal & Shantharajah (2015) employed the NSL-KDD dataset to detect anomalies based on network traffic patterns. In the study, the authors analyzed the relationship between the protocols and attacks used by the intruders to generate anomalous network traffic were related.

Table 2.2: Details on Normal and Attack Data in NSL-KDD Data set

Data set	Records	Normal Class	Dos Class	Probe Class	U2R Class	R2L Class
KDD Train+ 20%	25192	13449	9234	2289	11	209
		53.39 %	36.65%	9.09%	0.04%	0.83%
KDD Train+	125973	67343	45927	11656	52	995
		53.46 %	36.46%	9.25%	0.04%	0.79%
KDD TEST+	22544	9711	7458	2421	200	2754
		43.08 %	33.08%	10.74%	0.89%	12.22%

Creech & Hu (2013) claimed that the testing of new intrusion detection system algorithms against the datasets (KDD, UNM) was no longer relevant as the datasets were not representative of modern attacks. In 2013, the ADFA-LD dataset was made publicly available to aid the researchers to represent true performance against contemporary modern attacks (Creech &

Hu 2013). The ADFA-LD data set was published as an alternative to the widely used KDD98 dataset and was seen to contain low foot print attacks so that abnormal data become quite homogeneous and difficult to separate. The ADFA-LD data was collected using the Linux audit daemon.

The ADFA-LD dataset consists of 833 normal training sequences, 746 attack sequences and 4372 validation sequences, and has been used for evaluating system call based HIDS. The system call traces consists of call sequences represented as sequences of integers in the range 1 to 340. Due to the diverse and dynamic nature of system call patterns, it becomes difficult to separate the normal and abnormal behaviours in the ADFA-LD dataset.

Table 2.3: ADFA-LD Dataset

ADFA-LD		
Dataset	Traces	System Calls
Training Data	833	308,077
Validation Data	4,372	2,122,085
Attack DATA	746	317,388

2.2.2 ADFA-LD based Host based Intrusion Detection System

Forrest et al. (1996) proposed a sliding window based to extract a fixed size windows system call sequence as a trace. These system call traces are represented as a feature vector and have proved to be quite ineffective against handling long traces. Motivated by this, Kosoresow & Hofmeyr (1997) proposed another window frame based algorithm. The model determined the locality of anomalies within a trace by partitioning each trace into a number of small and fixed length sections called locality frames. This “window based” concept however resulted in a time consuming learning procedure.

Creech & Hu (2014) employed discontiguous system call patterns and claimed that original semantic feature based ELM (Extreme Learning Machine) turned out to be superior to all other 20 algorithms and obtained a Detection rate of 90% with 15% False Alarm rate but with the major drawback of a high computational time. Another framework by Borisaniya & Patel (2015) proposed Modified Vector space representation and utilized Vector Space Model with n-gram. The methodology represented system call traces as “bag of words”, which assigns weight to each and every word. For both binary and multiclass classification, the algorithm performed well but at the expense of high computational cost.

Later, Haider et al. (2015) attempted frequency based approach to implement an efficient kNN and k-means clustering (kMC) algorithm on ADFA-LD dataset. The concept considered frequency based model and reduced the dimensionality of feature vector system call traces. The model achieved a Detection rate (DR) of around 60% with an approximate 20% False Alarm rates (FAR). Further Miao modified the existing algorithm and transformed traces into short sequence based fixed length vector.

Marteau (2019) introduced the concept of an efficient algorithm (SC4ID), also known as Sequence Covering for Intrusion Detection system and achieved AUC of 0.842 using the kernel based family approach. However, the above stated kernel based methods proved inadequate to capture inter-word (system calls) relationships and sentence (system-call sequences) structure. In addition to the methodologies outlined above, sequence to sequence learning has achieved remarkable success in the field of machine learning tasks such as speech recognition, language models (Sutskever et al. 2014) and text summarization (Rush et al. 2015, Nallapati et al. 2016, Shen, Huang, Gao & Chen 2017) amongst others.

2.2.3 Sequence Anomaly Detection using Language Modeling

A language model for system call sequences specifies a probability distribution for the next call in a sequence given the sequence of previous system calls. A Neural Network is trained to produce this probability distribution using a training set of known normal sequences, that is, the network learns a language model of normal sequences.

The designed model as shown in Fig. 2.16 estimates the probability of a sequence occurring using these probability distributions. Note that $p(x_i|x_{1:i-1})$ is the probability of the integer x_i occurring after the sequence $x_{1:i-1}$.

$$p(x) = \prod_{i=1}^{l} p(x_i|x_{1:i-1}) \tag{2.23}$$

In practice the negative log of the value p(x) defined in equation 2.23 is used resulting in high values for unlikely sequences and low values for likely sequences. Anomaly detection for sequences can be carried out by imposing a threshold for this negative log likelihood (L) and predicting an anomaly for sequences with an L value above this threshold.

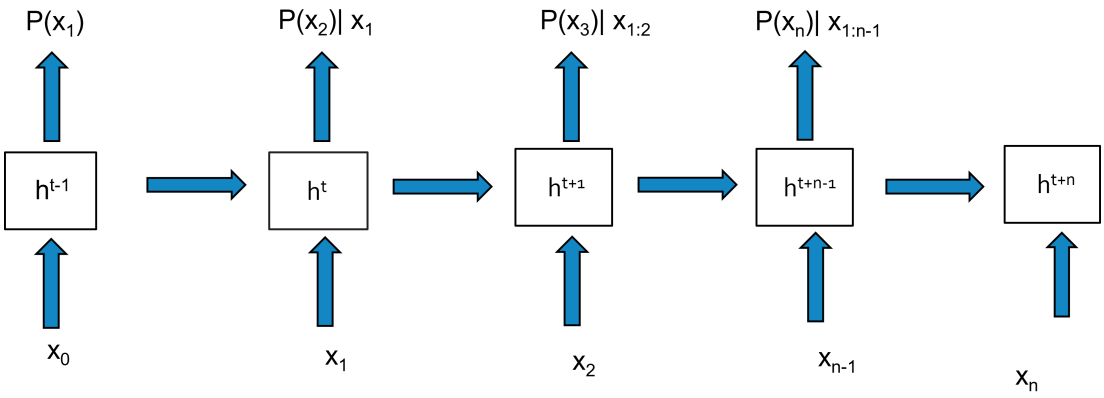


Figure 2.16: Conditional Probability over sequences of words

A Sequential Language based model calculates the probability distribution over the sequence of words. This concept has gained remarkable performance in terms of capturing inter word relationships. Work by Xie et al. (2014) illustrated a technical category that detected anomalies with a short sequence model using ADFA-LD dataset. The duplicate entries were eliminated thus forming the short sequences and achieved an acceptable performance of FPR of 20% and 70% accuracy. The authors represented the collaboration with the one-class SVM algorithm. The proposed model obtained a short sequence matrix by continuously transforming the training traces into the vectors of fixed-length. This process avoided unnecessary computing to eliminate the duplicated vectors occurring in the matrix.

Kim et al. (2016) illustrates a deep learning LSTM-RNN approach to evaluate an IDS model using KDD Cup 1999 dataset. The NSL-KDD data set consists of 4 different classes of attacks: Denial of Service (DoS), Probe, User to Root(U2R), and Remote to Local (R2L). The model was trained on the normalized dataset consisting of values from 0 to 1. The authors used True Detection Rate (TDR) and False Alarm Rate (FAR) as their proposed methodology metrics to evaluate the LSTM-RNN model. The experimental results from the model achieved 98% of detection rate with an average of 0.10% as false alarm rate. The LSTM-RNN model achieved the best overall model accuracy of 96.93% when compared to SVM, KNN, and Bayesian and among others. Additionally, the authors showed the impact of learning rate and hidden layer size to train an IDS model accurately.

Lv et al. (2018) proposed attention mechanism in combination with GRU based Encoder-Decoder mechanism for evaluating the performance of the anomaly detection algorithms. The model performed well when the long system call traces of ADFA-LD where the longer sequences were broken down into further smaller sequences. The length of sequence length varies from 10, 20 15, 18, 20, 22, 25 and 30. The HIDS achieved AUC of 0.94 with 90% True Detection Rate and 15% False Alarm Rate for predicted ROC. When compared to other models such as SVM, CNN and Random Forest, RNNs performed the best classifier.

Motivated by the RNN state-of-the-art, Chawla et al. (2018) proposed another computational efficient anomaly-based Host based Intrusion Detection System with a combined CNN/RNN Model. The proposed model comprises of CNN and GRUs substantially reduced training and testing time for the ADFA-LD dataset over LSTM based models. The stacked CNNs with GRUs model achieved 90% Detection Rate with 30% false alarm rate, resulting in an AUC value of 0.81 to detect malicious system calls. These results, in the cyber security domain, are one of the main contributions of the thesis.

Motivated by the LSTM work been done by Malhotra et al. (2015), Ding et al. (2019) proposed anomaly detection method for multivariate time series based on Long-Short Term Memory (LSTM) and Gaussian Mixture Model (GMM) model. Instead of conventional LSTM models, another variant of LSTM was proposed to improve the internal structure of LSTM to make it suitable for processing data time series anomaly detection. The proposed mechanism employed a multivariate Gaussian distribution to define correlations between the data attributes without loss of temporal performance. The experimental result from this model showed that the model achieved precision and recall values as 0.933 and 0.913 respectively. The joint detection of multiple dimensions to detect anomalies significantly reduces the false negative rate, thus improves the recall value and increases the value of F1 score. The authors mentioned that using this architecture, it is possible to improve the performance of the algorithm in the health system.

Sun et al. (2020) implemented a DL-IDS (Deep learning-based Intrusion Detection System), combination of CNN and LSTM for detecting the intrusion in the network traffic data. The authors used the concept of LSTM to extract the spatial and temporal network features. The proposed algorithm performed better on average low false alarm rates and high detection rates. The DL-IDS achieved 98.67% accuracy, which improved the state-of-the-art of detecting each attack type above 99.50% of designing a good intrusion detection system.

2.2.4 Reconstruction error based Anomaly Detection

In order to detect candidate anomalies, the autoencoder networks have been designed to train the normal instances and learn low-dimensional features to detect anomalous instances. Anomalous instances throw high reconstruction error when uses the bottleneck layers output to reconstruct the normal transactions of the original input data. There is much literature in this area that presents the solutions to detect anomalies.

Yousefi-Azar et al. (2017) proposed autoencoder-based feature learning for cyber security applications. They leveraged autoencoders to learn the semantic similarity among input features embedded in an abstract latent representation. The framework reduce the dimensionality of the features thereby significantly minimising the memory requirements. The single classifier model was able to perform malware classification and evaluate network-based anomaly intrusion detection. The authors used computationally efficient single classifier and the experimental results from multitask model achieved 84.12 % accuracy.

Interestingly, Zhou & Paffenroth (2017) demonstrated the idea behind eliminating the use of clean training data to detect the outliers. The novel extension to the deep autoencoder model discovered non-linear features and extracts the noise in the original data, inspired by Principal Component Analysis. Ribeiro et al. (2018) proposed the use of a convolutional autoencoder to detect the anomalous behaviour in video surveillance. The idea behind the implementation was to leverage the concept of reconstruction errors to measure the anomaly level of video frames. The AUC score of 0.847 evaluated the video data set classification with different thresholds.

2.2.5 Summary

The aforementioned papers have provided valuable insight about the current state-of-the-art. As demonstrated by the existing work above, several interesting proposals have emerged recently from the realization that DNNs work effectively in any domain application. Many works demonstrate that techniques using using recurrent neural networks perform better when compared to other traditional machine learning models. The work in this thesis intend to adopt this strategy to propose hybrid CNN and RNN based architecture that can learn meaningful information in a unsupervised manner.

In this thesis, a combined CNN/RNN based model (chapter 3) is discussed for anomaly detection in ADFA-LD system call trace sequential data. The CNN models are capable of extracting local information which are then fed as an input to the GRU layers, resulting in computationally efficient models. The key findings from this work would prove beneficial to many computer security incident response teams, which would benefit from frequent retraining to adapt to evolving typical usage.

The bidirectional autoencoder used in this thesis illustrates reconstruction error based approach. The proposed architecture demonstrates the capability to learn the sequence from both past and future directions. In addition to obtaining an optimal model for anomaly detection, the "**autoencoder**" mechanism has a second objective which is to support interpretation/introspection of the artificial neural network. This reconstruction error based approach forms the basis of an interpretability framework proposed later in the thesis. It allows for the identification of anomalous attributes and root cause identification only known to domain experts. In this way, troubleshooting anomalies across multidimensional data sets can be automated. The usefulness of the interpretation and its presentation can only be done by evaluation and feedback from security/network professionals.

2.3 Anomaly Detection in Telecommunications

This section begins with an overview of machine learning's evolution in telecom. This is followed by a discussion on detecting anomalies across univariate and multivariate scenarios. Finally, the section discusses the state-of-the-art anomaly detection approaches in telecommunications networks.

2.3.1 Radio Access Network

In a typical communication network, user equipments (UE), also known as wireless communication devices, mobile stations, stations (STA) and/or wireless devices, communicate via access networks such as a Radio access Network (RAN), to one or more core networks (CN). The RAN covers a geographical area which is divided into service areas or cell areas, with each service area or cell area being served by a radio network node such as an access node e.g. a Wi-Fi access point or a radio base station (RBS), which in some radio access technologies (RAT) may also be called, for example, a NodeB, an evolved NodeB (eNB) and a gNodeB (gNB) (*ETSI TS 133 511 V16.4.0* 2020). The service area or cell area is a geographical area where radio coverage is provided by the radio network node. The radio network node operates on radio frequencies to communicate over an air interface with the UEs within range of the access node. The radio network node communicates over a downlink (DL) to the UE and the UE communicates over an uplink (UL) to the access node.

Radio networks are influenced by many factors, both internal and external to the telecom network, and using isolated monitoring metrics on performance is not usually enough to indicate the true cause for failure. To gain a deeper understanding of causation requires a deeper investigation of other influencing factors, factors that are only known to domain experts.

A domain expert is required to identify precisely what is causing a problem in a communication network today. In network management Key Performance Indicators (KPI) are used to identify the existence of problems in a network. Typically, KPIs at this level are very high level and there is no indication of specificity about the problem. The KPIs may be used for rapidly detecting unacceptable performance in the network, enabling the operator to take immediate actions

to preserve the quality of the network, thus monitoring and optimizing the radio network performance. For example: monitoring the traffic flows, rates of failure, user connectivity, represents high level network issue but unable to identify the low level detailed about specific resources, ports, links etc. in the network.

Use of univariate anomaly detection is one approach to study or investigate what may be the cause of a KPI breach. Typically this is performed at a counter level where specific counters are targeted, and the univariate anomaly detection algorithm is customized and tuned per counter. Identifying the counters (features) to be investigated for specific KPI breaches is a manual task and tuning the algorithm in this case is also manual, which can lead to a lot of false positives, so the use of required post validation steps is essential.

2.3.2 Evolution of Machine Learning in Telecom

Telecommunication network technologies have evolved over the years from 1st generation (1G) to 5th generation (5G) (Vora 2015). From 1G telecom network technology to 4G technology, a significant evolution has happened in terms of features, functionalities, customer expectations, and operations and management paradigms.

The rising volume of network data due to the generation of approximately 1 million events every second in mobile RAN (Radio Network Access) data has contributed to increased momentum in network analysis and troubleshooting research (Al Mtawa et al. 2019). The importance of intelligent monitoring of network performance management is imperative for network operators. To ensure infrastructures provide a high level of robustness to customers, rule-based systems, based on domain knowledge, were implemented to analyze and detect the anomalies across multiple features. However, these traditional rule based network application approaches fail when exposed to new previously unseen complex patterns.This has highlighted the importance of anomaly detection and has resulted in drawing the researcher's attention towards adapting autonomous and intelligent network systems.

One important observation is that, in parallel to this telecom network system evolution, the application scenarios of statistics, machine learning and analytics have also evolved with significant changes. This parallel evolution in network technologies and analytics/machine learning is depicted in Fig 2.17, considering 2G/3G to 5G network technologies, together with the corresponding data analytics and machine learning techniques (Bosneag et al. 2016).

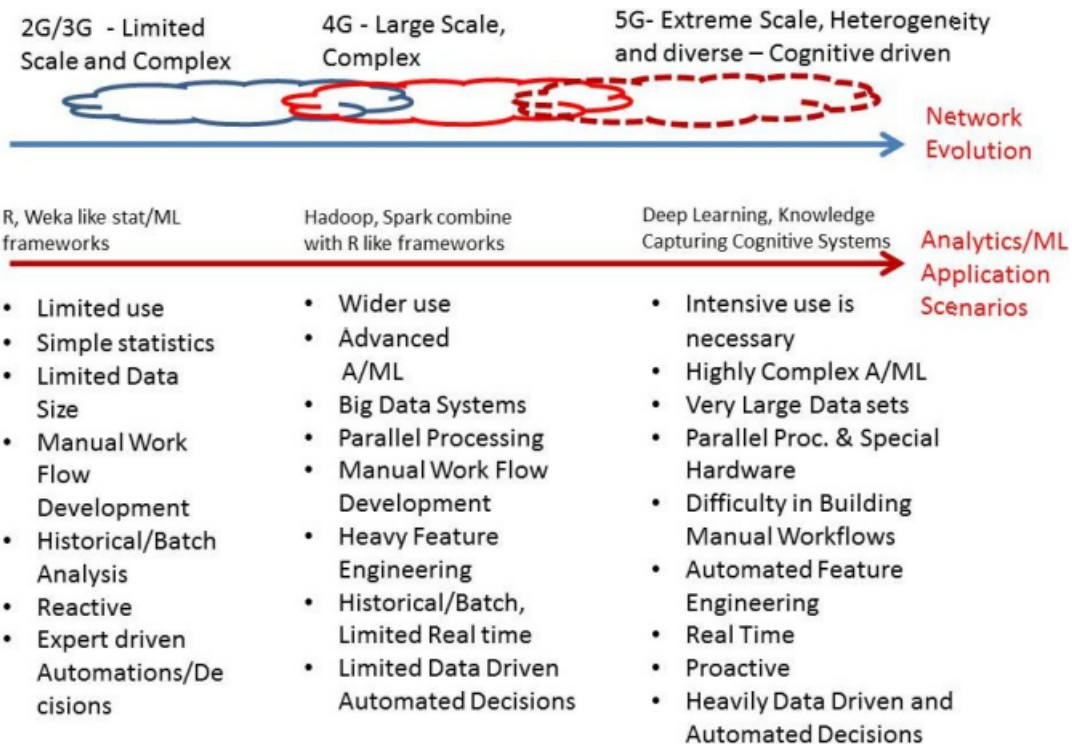


Figure 2.17: Parallel Changes in Network Technology Environments and Analytics/Machine Learning Application Scenario for Network Management

Deep learning algorithms are capable of processing and performing machine learning on large data sets efficiently. They can be applied to mine the very large data sets that are generated in large operator's networks and future 5G networks, and to find insights and other useful discoveries. Existing results show that deep learning algorithms are efficient in categorisation problems where the data space is extremely large, such as image recognition. In such situations, other machine learning mechanisms would be less efficient. Deep learning also improves the automation aspect, in that results are obtained with less supervision by the developer/data scientist. At the same time, the intrinsic layered structure of telecom networks, with higher level concepts and data being built from lower level ones, is also a natural fit for deep learning. All of this shows that deep learning is a technique that helps in handling complex problems with large data sets, and lends itself well to being applied to future telecom management systems.

Work conducted by Vittaldev et al. (2012) carried out various other predictor models such as Kalman Filter/Predictor, multi-layer perceptron model and CNN based sequence predictor. The best set of results were achieved using neural network reconstruction based predictors for detecting anomalies in radio network. Martinez et al. (2015) evaluated CADMANT (Context Anomaly Detection for Maintenance and Network Troubleshooting) technique for detecting abnormal patterns in unstructured textual data (logs/alarms). The approach proposed outlier detection without requirement of SQL/ Complex Event Processing queries. The CADAMANT framework overcomes the limitation of traditional structured query processing approaches. The authors provided an alternative free style query data approach to perform the fault detection in network data. The approach uses the concept of Distance based Outlier (DBO) detection mechanism. DBO involves a set of 1-Dimensional data stream contained within a collection of values, with maximum distance R of same term and Kmin nearest neighbors to cluster the data. The context factor algorithm incorporated the concept of sliding window with DBO to evaluate the anomalous behavior in the shortest period of time.

Recently many researchers have applied anomaly detection mechanisms (Fadlullah et al. 2017, Li et al. 2018, Abdullah Al Mamun & Valimaki 2018) to network data, and these have been quite successful in capturing anomalous behaviours. Wang & Handurukande (2018) empirically evaluated Autoregressive Integrated Moving Average (ARIMA). The engine takes time series data and further calculates the anomaly score and anomaly probability for a network traffic throughput dataset. Fernandez Maimo et al. (2018) proposed a novel self-adaptive deep learning framework for detecting anomalies in 5G RAN and EPC. An autoencoder based anomaly detector present in RAN quickly captures the anomaly symptoms, which is further evaluated by LSTM based model in Evolved Packet Core (EPC).

Trinh et al. (2019) proposed an approach to detect traffic anomalies using LSTM based RNN. The mobile network information is collected from LTE Physical Downlink Control Channel (PDCCH), which contains the radio scheduling information and has the benefit of being unencrypted and fine-grained, since the messages are exchanged every LTE subframe of 1 ms. The proposed framework scores F1 score of 1 by monitoring events from high concentration of people.

Flanagan et al. (2017) demonstrated a novel research on sliding window based clustering algorithm to work upon continuous 24-hour period Time-Series data. The proposal intended to capture the malicious network attacks present in a network through parallel clustering technique. The authors approached Micro-Cluster based Outlier Detection (MCOD) classifier to identify outliers over streaming data. As part of an anomaly detection algorithm, parallel clustering algorithms were presented, which worked concurrently to detect anomalies that might be lost while traversing time-series windows.

Al Mamun & Beyaz (2018) presented work on detecting anomalies in cellular mobile networks using LSTM based RNN. The novel methodology adopted by the authors for cell level performance profile generation leveraged the sequence learning capability of LSTMs to train the radio time series data (Call Drop Rate). The proposed framework calculates the error distribution using RMSE, 14.31 and 15.72 for training and test samples. This profile generation method is then used to recreate the actual profile and predict the profile of any cell for a given time series.

Fernandez Maimo et al. (2018) proposed a novel self-adaptive deep learning framework for detecting anomalies in 5G Radio Access Network (RAN) and EPC. The publicly available Czech Technical University (CTU) dataset comprises of unknown traffic collected from a large network with normal and background traffic (*Malware Capture Facility Project* 2013). An autoencoder based Anomaly detector present in RAN quickly captures the anomaly symptoms, which is further analysed by LSTM recurrent network model in EPC. The anomalies captured are further communicated to the Monitoring and Diagnoser Module (Garcia et al. 2014). The experimental results reaches the state-of-art classification accuracy in cyber defence architecture, hence optimizing the computing resources.

2.3.3 Summary

Currently, the existing works focus on examining traffic fluctuations to detect anomalies in cellular networks. The unsupervised technique (LSTM, Autoencoders) achieves state-of-the-art performance in detecting anomalies across the network. Fernandez Maimo et al. (2018) reiterated the fact that there is still a major challenge for developers to program their telecom services to leverage the benefits of DNNs.

Despite growing interest in deep learning in the mobile networking domain, existing contributions are not enough. It has been noted that existing AD work in telecommunications domain lacks the ability to explain the output of the model in a way that is intuitive for the end user. Motivated by this, chapter 4 addresses these issues by discussing the design of computationally efficient and interpretable DNN based models which forms the major contribution of the thesis. The design of a neural network model based novel architectures provides interpretation techniques. It provides insight and support for troubleshooting network data without any prior domain knowledge. The interpretable framework provides network operators with actionable insights for every anomalous network session, which enables a deeper investigation of influencing features. The unsupervised approach assists the analyst in identifying anomalies, categorizing them, and identifying their causes.

2.4 Interpretable Neural Networks

Normally an ANN is considered a ”black box” that cannot provide easily interpretable insights into the relationship between input and output. In particularly, when there are high dimensional data and layers, it becomes much harder to understand the reason behind an anomaly without a proper explanation.

Explainability and interpretability are often, in the context of machine learning, used interchanged. Hence, they are broadly defined as:

Explainability *is the extent to which the internal mechanics of a machine or deep learning system can be explained in human terms. An explanation usually relates the feature values of an instance to its model prediction in a humanly understandable way (Molnar 2019).*

Interpretability *is the degree to which humans can understand the cause of a decision/prediction (Molnar 2019). In other words, interpretability is about being able to discern the mechanics without necessarily knowing the maths behind calculating the neural network gradients.*

The field of eXplainable Artificial Intelligence (XAI) studies methods to provide explanations of the decisions taken by unintelligible, black box models, such as deep neural networks (Moradi & Samwald 2021), (Guidotti et al. 2019). Explainable AI, also known as explainable artificial intelligence, XAI or interpretability, is a research field related to trust and transparency of AI-based systems. XAI aims to create trust between machines and humans and create transparent AI that would explain its decisions to its users. In the last few years, many researchers have investigated the area of explainability of AI, which has become a major concern for many application sectors.

Barredo Arrieta et al. (2020) explained the importance of concepts, taxonomies, opportunities and challenges toward practical deployment of AI models. The paper emphasizes the importance of eXplainable Artificial Intelligence (XAI), explain model predictions in real-life applications. The paper illustrated the contributions related to the explainability of different Machine Learning models, describing different post-hoc explainability approaches.

Montavon et al. (2018) provided an entry point to the problem of interpreting a deep neural network model and explaining its predictions. They focused on layer-wise relevance propagation (LRP) technique, to provide the interpretations on real data. The authors demonstrated the machine learning the model's transparency decisions by identifying the relevant input variables. The authors discussed the effect of linking prototypes to the data using a data density function with graph structure technique.

Model interpretation mainly consists of subjective reasoning from the perspective of an end user or a domain expert. It plays an integral part of how the model evaluations are displayed. Shen et al. (2020) facilitated the model interpretability by leveraging the hidden latent space of a stacked trained LSTM autoencoder model by employing t-Distributed Stochastic Neighbor Embedding (t-SNE). Parallel coordinate plot and heatmaps were generated to identify the essential music elements (features) to understand the value distribution of latent representations by comparing multiple features for a collection of music representation.

Carletti et al. (2019) proposed a novel Depth-based Feature Importance for the Isolation Forest (DIFFI), a method to provide interpretation traits using Isolation Forest (IF) algorithm. The algorithm emphasized on the feature importance required for anomaly detection algorithm. Their algorithm computational cost has been compared to the state-of-the-art SHAP with significant lower computational costs.

2.4.1 Post-hoc Interpretation

This section focuses on the problem of interpreting a concept learned by a deep neural network (DNN). Post-hoc explanations refer to analyzing models to offer explanations after the model is trained. There is a need for a method (post-hoc, after training) which can be used to provide the interpretability of models on the datasets, where the domain knowledge is not available. Post-hoc interpretation can further be divided into two categories:

(i) **Global Interpretation**: Global methods describe the average behavior of a machine learning model. This level of interpretability is about understanding how the model makes decisions, based on a holistic view of its features and each of the learned components such as weights, other parameters, and structures. A global explanation describes the working process of the ML method overall. This approach helps to understand the distribution of the target outcome based on the features.

(ii) **Local Interpretation**: Local interpretation methods explain individual predictions. Local interpretation focuses on specifics of each individual and provides explanations that can lead to a better understanding of the feature contribution in individuals and smaller groups that are often overlooked by the global interpretation techniques.

2.4.1.1 Model-Agnostic Methods

Local model-agnostic explanation methods aim to explain a single prediction of an arbitrary machine learning model by studying the model through its inputs and corresponding outputs. The neural network based models are capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions. It would be beneficial to know why a model gives a certain prediction, which Local Interpretable Model-Agnostic Explanations (LIME) aims to solve. Ribeiro et al. (2016) proposed LIME methodology. Explaining the predictions of a complex machine learning model helps practitioner to debug the model and build user's trust in the predictions.

LIME aims to make the predictions of the neural network model interpretable at the same time by perturbing the input of data samples to observe the resulting impact on the output and how the predictions interact with it. The output of LIME is a list of local interpretations, reflecting the contribution of each feature to the prediction of a data sample. The approach works well with linear models to approximate local behaviour. To a certain extent, this assumption is correct when looking at a very small region around the data sample. By expanding this region however, it is possible that a linear model might not be powerful enough to explain the behavior of the original model. Non-linearity at local regions happens for those datasets that require complex, non-interpretable models. Not being able to apply LIME in these scenario's is a significant pitfall. LIME generates the prediction probabilities, which still results in poor model interpretability visually, and still makes it hard for the end-user to understand.

Lundberg & Lee (2017) proposed a relatively new technique in machine learning known as SHapley Additive exPlanation (SHAP) to overcome the visualization interpretability of the model output. SHAP supports the interpretation of the neural network, or any complex machine learning model, by determining how input features contribute to the value of output features.

2.4.1.2 SHapley Additive exPlanation (SHAP)

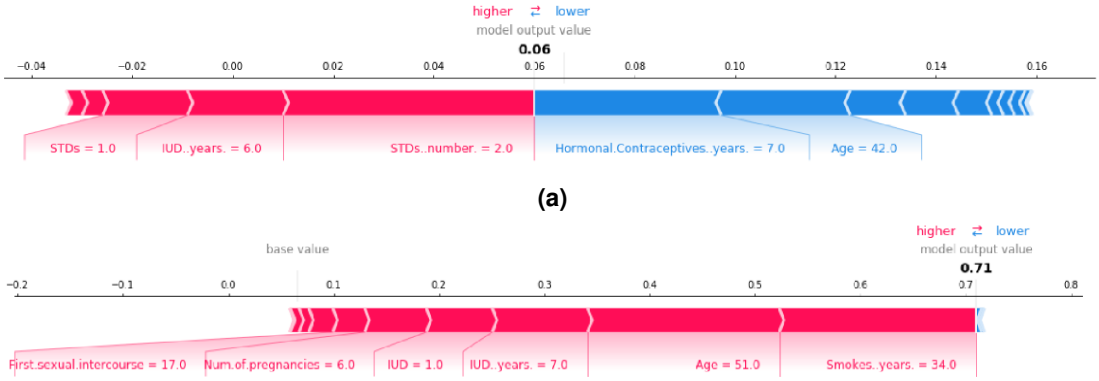
A game theory-based framework known as SHapley Additive exPlanations (SHAP) combined several existing explanation methods into one class of additive feature attribution methods (Shapley 2016). The key idea of SHAP is to calculate the Shapley values for each feature of the sample to be interpreted, where each Shapley value represents the impact that the feature to which it is associated, generates in the prediction.

The Shapley value is the average contribution of a feature value to the prediction in all possible coalition defined as:

$$\phi(m,x) = \sum_{z' \subseteq x'_1,\ldots,x'_n} \frac{|z'|!(M - |Z'| - 1)!}{M!} \cdot [m(z^i \cup x'_i) - m(z')] \tag{2.24}$$

where z' is a subset of the features used in the model, x is the vector of feature values of the instance to be explained. M is the number of features, m(z') is the prediction for feature values in set z'. When calculating m(z'), the ith feature is masked out and then simulated by drawing random instances or the random values of the ith feature from the dataset. Calculating the Shapley value requires computation time and power since there are 2^k possible coalitions of the feature values. One of the advantages of using SHAP is that it can distribute the difference between the prediction and the average prediction evenly distributed among the features of the instance's feature values because of the Shapely values efficiency characteristic.

Force plots in SHAP are very intuitive and visual way to understand the influence of each feature on the models prediction for a specific instance of the data. For example, Fig 2.18 shows the SHAP values to explain the predicted cancer probabilities of two individuals. The baseline – the average predicted probability – is 0.066. The first woman has a low predicted risk of 0.06. Risk increasing effects such as STDs are offset by decreasing effects such as age. The second woman has a high predicted risk of 0.71. Age of 51 and 34 years of smoking increase her predicted cancer risk.



(a)

(b)

Figure 2.18: (a) SHAP values to explain the predicted cancer probabilities of two individuals. The baseline – the average predicted probability – is 0.066. The first woman has a low predicted risk of 0.06. Risk increasing effects such as STDs are offset by decreasing effects such as age. (b) The second woman has a high predicted risk of 0.71. Age of 51 and 34 years of smoking increase her predicted cancer risk.

2.4.1.3 Kernel SHAP

Kernel SHAP is an efficient model agnostic method that allows the calculation of Shapley values with much fewer coalition samples. Kernel SHAP uses a special weighted linear regression to compute the importance of each feature. It is a model agnostic method to approximate SHAP values using the ideas from LIME and Shapley values. The Shapley kernel that recovers SHAP values is given by:

$$\pi_{x'}(Z') = \frac{(M-1)}{(M\ choose\ |Z'|)\ |\ Z'\ |(M\ -|Z'|)'} \tag{2.25}$$

Where M is the number of features & |Z'| is the number of non-zero features in the simplified input Z'.

2.4.1.4 Deep Explainer SHAP

Deep Explainer SHAP is an enhanced version of the DeepLIFT algorithm (Deep SHAP), which approximates the conditional expectations of SHAP values using a selection of background samples. Lundberg & Lee (2017) showed that the per node attribution rules in DeepLIFT can be chosen to approximate Shapley values. Deep SHAP is a high-speed approximation algorithm for SHAP values in deep learning models. Deep SHAP estimates approximate SHAP values such that they sum up to the difference between the expected model output on the passed background samples and the current model output (f(x) - E[f(x)]).

2.4.1.5 SHAP for Autoencoders

Antwarg et al. (2021) introduced the idea of providing interpretation of neural network models. They demonstrated the idea of providing interpretation to neural network model. The authors emphasized the importance of how hard it is to explain the anomalies extracted by anomaly detection algorithms. The authors extended SHAP to explain anomalies detected by an autoencoder, an unsupervised model. The authors proposed a black-box interpretable method which aims to provide a comprehensive root cause analysis. In the framework, the top features with high reconstruction errors were connected to the features that contribute most to reconstruction error. This detailed interpretation of why an instance is anomalous enables the domain experts to focus their investigation on the most important anomalies and increase their trust in the algorithm.

Motivated by this, Chawla et al. (2020) emphasized on the importance of explaining the anomalies extracted by an autoencoder based anomaly detection algorithm. The paper provided an illustration to illustrate the Kernel SHAP explainer framework and formed the basis of major contribution of thesis. It can extract and visually depict the instances and the top contributing features with the highest reconstruction errors. The illustration in the paper provided a framework to explain anomalies identified by an autoencoder without knowing the underline architecture of the autoencoder model. The Force plot method of SHAP is a visual intuitive method which provides a better method to understand the anomalies (Lundberg et al. 2018). Hence, making the complex “black-box” models more interpretable. SHAP provides a force plot to visually explain the features that contribute the most to give an insight into the output. The results from SHAP are one of the major contributions of the thesis.

2.5 Subspace Clustering

L. et al. (2004) illustrated the need for subspace clustering by creating a sample data set with 400 instances in 3 dimensions. Fig 2.19 shows the data distribution comprised of the first two clusters in dimensions a and b. The data is of a normal distribution with μ 0.5 and -0.5 and σ = 0.5, 0.2 in dimension a and b respectively. In dimension c, these clusters have μ = 0 and σ = 1. The second two clusters are in dimensions b and c and were generated in the same manner.

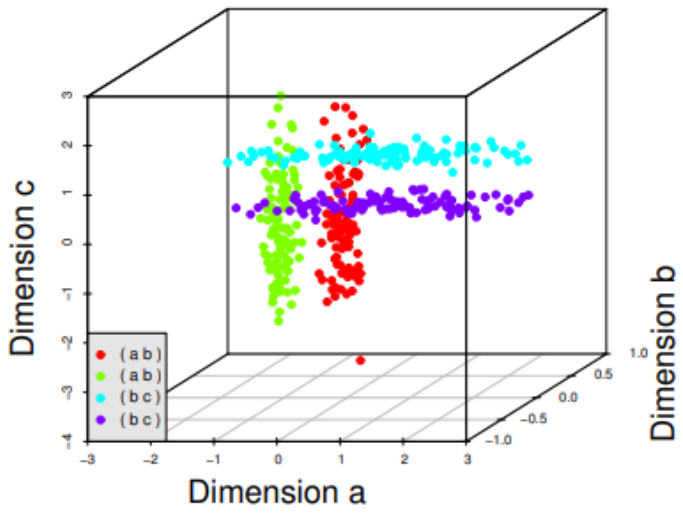


Figure 2.19: Example of Subspace Clustering with four clusters

When k-means was applied to cluster these data points, it did not perform well to find the relevant set of clusters. This is because each cluster is spread out over some irrelevant dimension. In higher dimensional data sets this problem becomes even worse and the clusters become impossible to find, suggesting that considers fewer dimensions.

So, the traditional clustering algorithms assume a single pattern and/or does not look for subspaces. In those scenarios, the concept of Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, possibly overlapping subspaces using Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP) algorithm (You et al. 2016).

Given a set of points in D dimension, X = $\{x_j \in R^D\}_{j=1}^N$, lies in unknown number of subspaces $\{Si\}_{i=1}^n$ of unknown dimensions $d_{i i=1}^n$. Subspace clustering mechanisms group data into groups such that the group contains only data points from the same subspace. OMP based subspace clustering method computes the self-representation matrix C via solving the following elastic net problem. Here, X =$[x_1, ..., x_N] \in R^{D \times N}$ is the input data matrix and subspace preserving $C = [c_1, ..., c_N] \in \mathbb{R}^{N \times N}$ is the outputted matrix of coefficients.

$$c_j^* = arg\ min||x_j - X_{c_j}||_2^2 \tag{2.26}$$

$$||c_j||_0 \leq k,\ c_{jj} = 0 \tag{2.27}$$

where the vector $c_j^* \in \mathbb{R}^N$ is the jth column of $C^* \in \mathbb{R}^{N \times N}$ computed as OMP $(X_{-j}, x_j) \in \mathbb{R}^{N-1}$

The subspace clustering is an extension of traditional N dimensional cluster analysis which allows to simultaneously group features and observations by creating both row and column clusters. Macha & Akoglu (2018) explained anomalies via the x-PACS algorithm for explaining Patterns of Anomalies with Characterizing Subspaces by unearthing their hidden characteristics. Explaining anomalies in groups saves analyst time and gives insight into different aspects of anomalies.

The application of DNNs in anomaly detection achieves the state-of-the-art, but the end user cannot easily interpret their output. Motivated by the subspace clustering work done by You et al. (2016), the CCA architecture (chapter 4) leverages the subspace clustering to bring effective interpretation to address the research question 2 (1.2). This thesis contribute towards the interpretation of an unsupervised model, which in turn will be of great benefit to any application domain analysts.

2.6 Performance Evaluation

The general process used to determine optimal models is described in the Cross Industry Standard Process for Data Mining (Crisp-DM) (Shafique & Qaiser 2014). The steps involved are business understanding, data understanding, data preparation, modeling, evaluation and deployment. Many iterations are possible and indeed desirable before deployment. In particular model evaluation can lead to a reassessment of the business process (cyber security) and the data, as well as repeated iterations of model building in order to obtain the optimal meta-parameters for the model. In the area of ANNs, considerable experience and expertise needs to be and will be developed in order to optimally configure a neural network.

In order to train a model for anomaly detection it is necessary to train it on normal data. Otherwise the model learns from attack data and in effect becomes a signature based method and is unable to identify so called zero-day attacks. Data sets normally contain normal and attack data. The normal data is split into training data and test data. The attack data is only used for testing. Models are trained on normal data and are evaluated by their performance in classifying test data as normal or attack data.

2.6.1 Performance Metrics

As this is a binary classification task (though unsupervised), model performance is evaluated on metrics that depend on the number of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). Normally models have a parameter, often a probability, so that when the threshold for this parameter is changed, the number of predicted positives and negatives change. In effect this gives the user of the model a way of tuning it depending on cost associated with false positives (FP) and false negatives (FN). The entries in the confusion matrix are denoted as:

- **True Positive (TP)**: This entry refers to the number of positive examples which are correctly predicted as positives.
- **True Negative (TN)**: It denotes the number of negative examples correctly classified as negatives.
- **False Positive (FP)**: This entry is defined as the number of negative examples incorrectly classified as positives.

- **False Negative (FN)**: It is the number of positive examples incorrectly assigned as negatives.

In order to evaluate a model without reference to this threshold chosen, a ROC curve is obtained by plotting True Positive Rate (TPR) v False Positive Rate (FPR) for the range of threshold values. The area under this curve is a measure of the performance of the model. The main motivation for this work is to implement deep neural architectures that yield improved performance for intrusion detection systems as measured by these metrics. The metrics are defined as follows:

- **True Positive Rate/ Recall /Sensitivity**: Probability that an observation is classified as anomaly when in fact it is an anomaly.

$$DR = \frac{TP}{TP + FN} \tag{2.28}$$

- **False Positive Rate:** Probability that an observation is classified as an anomaly when in fact it is a regular observation.

$$FPR = \frac{FP}{FP + TN} \tag{2.29}$$

- **Specificity**: Specificity is defined as the proportion of actual negatives, which got predicted as the negative (or true negative). This implies that there will be another proportion of actual negative, which got predicted as positive and could be termed as false positives.

$$Specificity = \frac{TN}{TN + FP} \tag{2.30}$$

- **Precision**: which indicates what percent of positive predictions were correct.

$$Precision = \frac{TP}{TP + FP} \tag{2.31}$$

- **Accuracy**: Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions the model got right (*Classification: Accuracy* 2022). Formally, accuracy has the following definition:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.32}$$

- **Area under Precision-Recall**: The area under the precision-recall curve (AU-PR) is a model performance metric for binary responses that is appropriate for rare events and not dependent on model specificity (Davis & Goadrich 2006).

- **Receiver Operating Characteristic**: Receiver operating characteristic (ROC) is "*a method of quantifying how accurately experimental subjects, professional diagnostics and prognosticators (and their various tools) perform when they are required to make a series of fine discriminations or to say which of two conditions or states of nature, confusable at the moment of decision, exists or will exist*" (Hanley et al. 1989).

- **Area Under the Curve (AUC)** - ROC curve is a performance measurement for the classification problems at various threshold settings. The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. Each point on the ROC curve represents a TPR/FPR pair corresponding to a particular decision threshold.

  The area under the ROC curve or AUC score is used for the multiclass or binary classification problem, and it demonstrates how the model is discriminating positive and negative classes. The ROC-AUC score can be a useful performance metric, particularly if the value of positive and negative classes is equal for classification.

  As shown in Fig. 2.20, Hanley et al. (1989) obtained ROC curve for diagnostic accuracy to distinguish between the two states. The different set of statistical tests conducted for patients resulted in calculating TP, TN,TP and FN.

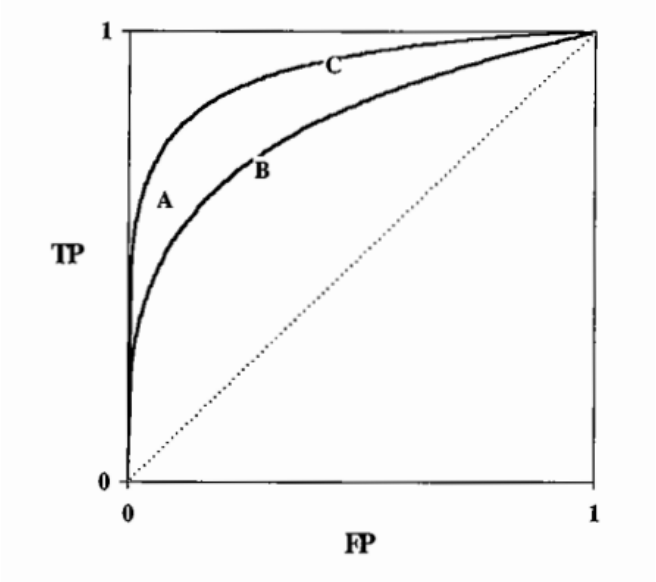


Figure 2.20: Receiver Operating Characteristic Curve depicting three diagnostic test curves.

The plot of TP fraction as a function of the FP fraction is called as a ROC curve. The pair of TP and FP values from each diagnostic study has been plotted as two dimensional data points on FP:TP axes. Likewise, as a result of different studies been conducted, the data points from the different studies of a test were plotted on a single ROC curve. This illustrates the test capacity to discriminate a given disease from non disease, reflecting different choices of decision criterion.

The ROC curve shows test C has poorer specificity (higher FP) than test A. It uses a pair of statistics – true positive rate and false positive rate – to characterize a classifier's performance. Test B has higher sensitivity than test A, but comparatively poorer discriminability than test A and C. Hence, ROC analysis is a graphical approach for analyzing the performance of a classifier (Tan 2009).

- **Confusion Matrix**: A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

Confusion Matrix

actual value	p	n	total
p′	True Positive	False Negative	P′
n′	False Positive	True Negative	N′
total	P	N	

In addition to obtaining an optimal model for anomaly detection, “autoencoder” based anomaly detection solutions has a second objective which is to support interpretation/introspection of the ANNs. This enables the identification of responsible features for the classification of anomalies. The usefulness of the interpretation and its presentation can only be done by evaluation and feedback from security and telecom network professionals.

Chapter 3

Anomaly Detection for Sequential Data

This chapter outlines two approaches to anomaly detection for sequential data on Cyber Security. The first is a prediction based approach where the design of a computationally efficient CNN/RNN based neural network solution is proposed with reduced training and testing times. The second is a reconstruction error based approach where the definition of a bidirectional autoencoder based anomaly detection algorithm implements a Host Based Intrusion Detection System and classify anomalous sequences of operating system calls.

The goal of this chapter is to define an approach that achieves optimal classification accuracy and improved training times. The techniques are applied to the problem of Host Based ID, where Forrest et al. (1996) suggested that sequences of system calls could be used to capture normal behaviour in a computer system. Fig. 3.1 shows the flowchart for evaluating HIDS. These experiments will answer the first research question.

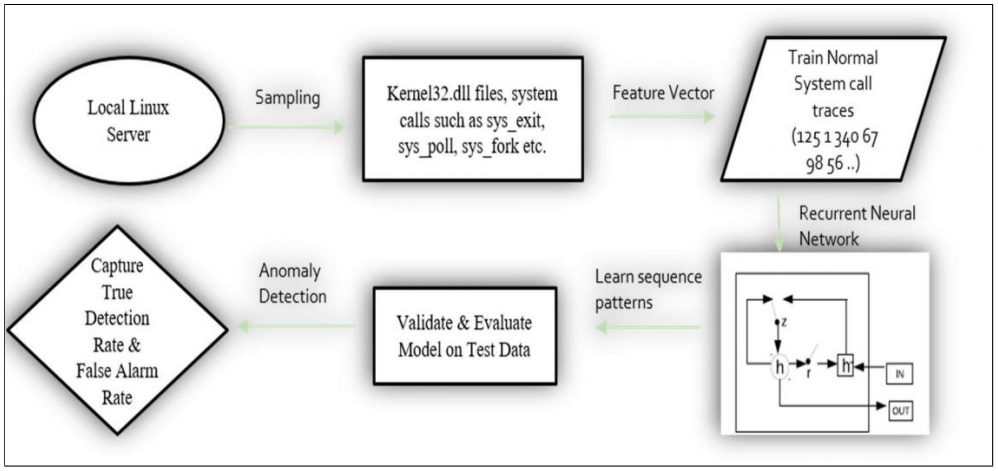


Figure 3.1: Flowchart for evaluating HIDS

Research Question 1: *Is it possible to design and develop neural network based anomaly detection solution with reduced training and testing times, and get optimal anomaly detection results for sequential data ?*

3.1 Prediction based approach to AD for sequential data

The first approach to AD for sequential data is a prediction based which determines the probability of a particular call sequence occurring from a language model trained on normal call sequences. The first paper titled *"Host based Intrusion Detection System with Combined CNN/RNN Model"* classifies the low probability occurring sequence as an anomaly (Chawla et al. 2018).

Over the past few years, seq2seq learning has achieved remarkable success in the field of machine learning tasks such as speech recognition, language models (Sutskever et al. 2014, Chorowski et al. 2015) and text summarizing (Rush et al. 2015, Nallapati et al. 2016) amongst others. This type of learning requires to train a neural network model to predict the next element in a sequence. Convolutional Neural Networks (CNNs) were shown to perform well on certain sequence processing problems at a considerably cheaper computational cost than Recurrent Neural Networks (RNNs). Wang et al. (2016) illustrated that the combined architecture of CNN-RNN as described in dissemination technique was able to achieve high accuracy for sentiment analysis in shorter text.

Motivated by these applications in the domain of Deep Neural Networks, an architecture was proposed with two significant contributions. Firstly, to model sequence to sequence learning which is a combination of a multilayer CNN with an RNN made up of Gated Recurrent Units (GRUs) where local features in the input sequences are extracted by the CNN layer and used as an input to the GRU layer. The output from the GRU layer is processed with a fully connected softmax layer that outputs a probability distribution over system call integers. It was demonstrated that GRU was capable of effectively replacing LSTM and obtaining similar results with reduced training times. To the best of knowledge none of the previous work has focused on stacking CNN with GRU in a security domain to develop IDS.

3.1.1 Architectural Design & Methodology

The model architecture as shown in Fig. 3.2 consists of a number of layers.

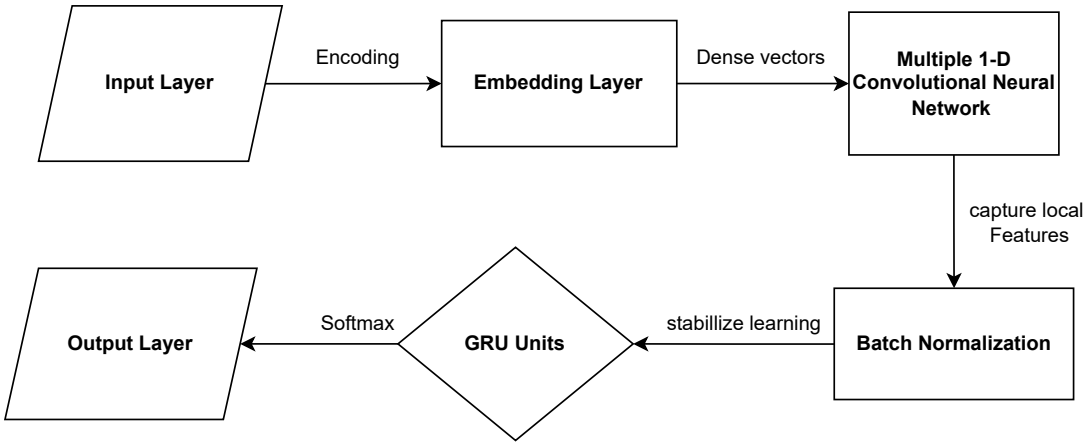


Figure 3.2: HIDS Model Architecture

The embedding layer performs word embedding and transforms one hot encoding of integers in the call sequence, which vary from 1 to 340, into a dense vector of size 32. Embedding layer weights are learned during training, that is a pre-trained embedding is not used. Embedding layer is the first dense layer in the neural network, where the output vectors obtained are of low dimensionality (François Chollet 2018). This concept has been generally derived from the concept of NLP related tasks (Li & Yang 2018). Generally, one hot encoding of input vectors is sparse with very high dimensional output. Like any other dense layer in a neural network, the word embedding layer is also trainable, where each input integer is used as the index to access a table that contains all possible vectors. Hence, it results in a lower dimensional output vector.

Here, the columns in Fig.3.3 represent the sparse one-hot encoded vectors (words) and the length of the column represents the number of dimensions used to represent the word (dense embedded representation). The geometric relationships between word vectors should reflect the semantic relationships between these words. Word embeddings map human language into a geometric space. For instance, in a reasonable embedding space, expect synonyms to be embedded into similar word vectors.

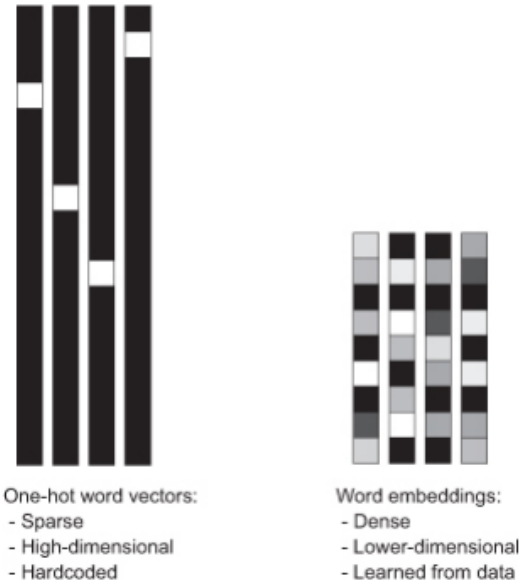


Figure 3.3: Embedding: Whereas word representations obtained from one-hot encoding or hashing are sparse, high-dimensional, and hard coded, word embeddings are dense, relatively low dimensional, and learned from data

The 1D CNN layer processes the input batches and extracts local patterns from a sequence. RNNs are a powerful tool for modeling sequential data, but the dependence of each timestep's computation on the previous timestep's output limits parallelism and makes RNNs unwieldy for very long sequences. Thus, 1D convolutions were incorporated to perform scalar multiplications and additions with RNNs to speed up the model training process.

The input system call sequences is sent in minibatches firstly to be processed by CNN layers and then further sent to GRU for sequential learning. Such 1D convnets can be competitive with RNNs on certain sequence processing problems, usually at a considerably smaller training times, resulting in computational efficient model. Thus, 1D convnets nets are used as a pre-processing step to make the sequence smaller resulting in a faster training. In practice, the CNN layers extract higher level local features, which are then passed on to the GRU as an input.

The Batch Normalization layer helps with gradient propagation and significantly increases the training speed. The GRU layer, with a Keras parameter “return sequences” set to true returns the hidden state output for each input time step and is necessary when passing data to the TimeDistributed Layer. A TimeDistributed Dense layer allows the build of models that have a one-to-many or many-to-many architecture (Dong et al. 2015).

The Time Distributed Dense model applies the same dense layer to every time step during GRU Cell unrolling. This is because the output function for each of the "many" outputs must be applied to the same function at each timestep. The TimeDistributed Dense layers allow the Dense function to be applied across every output over time. In Keras, the TimeDistributed layer applies a layer to every temporal slice of an input (François Chollet 2018). This Dense layer applies the same Dense (fully-connected) operation to every timestep of a 3D input vector.

3.1.2 Algorithm for Prediction based AD

Given a set of normal sequences, the anomaly detection algorithm evaluates the test system call trace as anomalous or normal. As the algorithm implemented is cost efficient (reduced training and testing times), it consumes much less computational resources in comparison to the traditional deep neural network algorithms. The following steps specify the algorithm for anomaly detection and its evaluation on validation data.

1. **Training**
    - The RNN based model is trained in minibatches on normal call sequences in the ADFA-LD dataset (no attack sequences).
    - The model is trained in minibatches with the set of normal system call traces (no attack sequences) ADFA-LD dataset. The loss function is the categorical cross entropy function.

2. **Calculation of Sequence Probability**
    - The input sequence is fed through the trained model. The output is a sequence of probability distributions using hidden states, which represent the probability distribution for the next integer in the sequence.

$$p(x) = \prod_{i=1}^{l} p(x_i | x_{1:i-1}) \tag{3.1}$$

- A sequence probability value is calculated by essentially multiplying the probabilities of the next integer in the sequence occurring, across the entire length of the sequence.

3. **Identification of anomalies**
    - The negative log of the sequence probability is calculated for every sequence in the validation data.
    - If the negative log value for the sequence is greater than the threshold (In an unsupervised environment, the threshold is chosen as two standard deviations below/above the mean), the sequence is classified as attack (Positive), otherwise it is classified as normal (Negative).
    - The sequence is identified as either TP, FP, TN, FN.
    - The DR and FAR for the particular threshold value is calculated.

4. **Calculation of ROC curve data and AUC for validation data**
    - The ROC curve is plotted for the range of threshold values.
    - The AUC value is calculated for the ROC curve.

3.1.3 CNN/GRU Based Experimental set-up

For the purposes of experiment evaluation, five independent models were designed, which comprised of:

1. One layer with 200 GRU units.
2. One layer with 200 LSTM units.
3. Six layered 1D CNN with 200 GRU units.
4. Seven layered 1D CNN with 500 GRU units.
5. Eight layered 1D CNN with 600 GRU units.

Each model was trained with 833 normal sequences, which were processed in variant length mini batches, where each sequence in a mini batch was padded to the length of the longest system call in the mini batch. The model was trained using Adam optimizer (have faster computation time, and require fewer parameters for tuning) with a learning rate of 0.0001, a softmax activation function in Time Distributed layer and relu activation function at the CNN layer with drop out probability of 0.7 before the softmax layer.

The TimeDistributed Layer applies a same Dense (fully-connected) operation to every timestep of a 3D input and allows us to gather the output at each timestep, effectively supporting sequence to sequence learning. The output layer is a Keras Dense layer, essentially a regular densely connected neural network layer. It is used with a softmax activation function in order to predict a probability distribution for the next integer in the call sequence.

3.1.4 Experimental Results

The specification of the computational machine includes Intel core i7-8700@3.20GHz processor, 16GB of RAM and NVIDIA GeForce GTX1070 GPU running 64-bit Windows 10 operating system and the NVIDIA CUDA Deep Neural Network library (CuDNN). The Keras python library was used running on top of a source build of TensorFlow 1.7.1 with CUDA support (François Chollet 2022).

Equation 2.23 was used to calculate an overall probability for the sequence and Fig.3.4 shows the ROC curves for the above outlined models. The model with CNN+GRU 600 units gave the best value (0.81) for the Area Under the ROC curve (AUC). CNN+GRU 200 and 500 units were only marginally behind resulting in an AUC value of 0.80. The model produces 90% True Detection Rate with a False Alarm Rate of 30%. CNN-GRU model improves its performance when comparing with the individual GRU and LSTM based models.

3.1.5 Discussion on Prediction based Approach for AD

The model is trained on normal call sequences of variable length and predicts a probability distribution for the next integer in a call sequence. This in turn is used to predict a probability for the entire sequence and a threshold for classification is chosen from the range of negative log likelihood values. The proposed CNN-GRU language model have maintained near state-of-the-art performance for neural network models with a substantial reduction in training

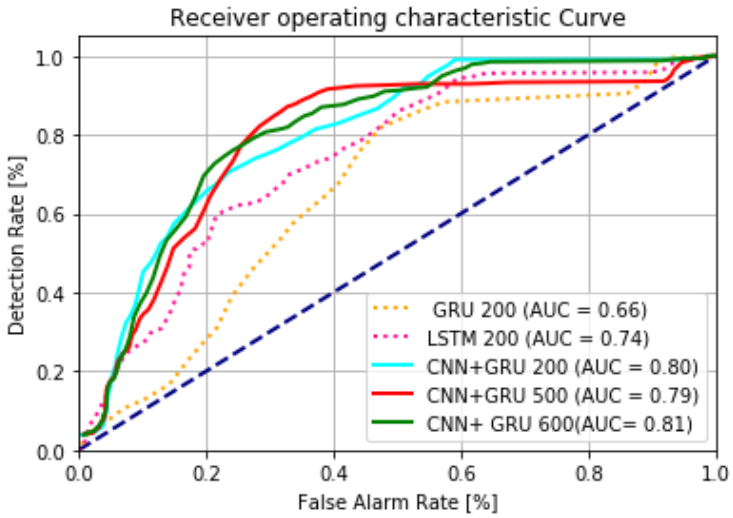


Figure 3.4: ROC curve comparing different models of ADFA Dataset

times compared to LSTM models. In the proposed model, 1D CNN layer processes the input batches and extracts local patterns from a sequence. RNNs are a powerful tool for modeling sequential data, but the dependence of each timestep's computation on the previous timestep's output limits parallelism and makes RNNs unwieldy for very long sequences. Thus, 1D convolutions were incorporated to perform scalar multiplications and additions with RNNs to speed up the model learning process.

The experimental results have shown that the CNN-GRU language based neural network model has substantially reduced training time when compared to an LSTM model. Additionally, the use of large mini-batches as an input to the model increases the available computational parallelism, thereby reducing the overall training time. Secondly, it achieves better accuracy by stacking multiple CNN layers before the GRU layer. The time taken for a stacked CNN/GRU model is approximately **167.7%** times faster than LSTM due to faster convergence in training. While CNN-GRU based model converged after 10 training epochs, giving an AUC of 0.80, comparatively the LSTM based model needed 100 epoch to converge resulting in an AUC of 0.74.

Lv et al. (2018) were able to achieve an AUC of 0.94 with 90% True Detection Rate and 15% False Alarm Rate. They trained their model using 833 sequences, breaking longer sequences down into smaller sequences. To improve the performance of baseline intrusion detection classifiers, the authors combined the predicted sequence as a supplementary part with in-

voked sequence to form an extended input. The experimental results evaluated that BLEU score is positive proportion to the input length, which means that the prediction model obtains more effective information by increasing the length of the input sequence. However, when the sequence length continues to grow, the BLEU value declined due to the limitation of GRU memory. The system-call back position in the sequence dilute the information of the previous system calls causing the loss of information, and then the prediction errors increased. Thus, when using the model, an appropriate length of system calls is required. As compared to the proposed model, there was no breakdown of the training sequences, as this was handled by incorporating 1D CNNs (Chawla et al. 2018).

Marteau (2019) demonstrated the concept of an efficient algorithm (SC4ID), also known as Sequence Covering For Intrusion Detection system and achieved AUC of 0.842 using the kernel based family approach for ADFA-LD dataset. However, the above stated kernel based method proved inadequate to capture inter-word (system calls) relationships and sentence (system-call sequences) structure. In comparison, the proposed CNN/GRU model, 1D CNNs extracts the local features and capture the inter-word relationship. As shown in Table 3.1, the proposed model was able to achieve 90% True Detection Rate and the False Alarm Rate of 30% using a combined CNN/GRU model with 0.81 AUC while training 833 sequences of variable length.

Table 3.1: CNN/RNN based Model Analysis

Model	RNN Units	Training Time (sec)	Testing Time (sec)	AUC
GRU	200	376	444	0.66
LSTM	200	4444	541	0.74
CNN+GRU	200	390	441	0.80
CNN+GRU	500	402	493	0.79
CNN+GRU	600	413	533	0.81

Thus, the computationally efficient anomaly based intrusion detection relies on determining the probability of a particular call sequence occurring from a language model trained on normal call sequences. The combined CNN/GRU model extract features from the input minibatches to recognize the local patterns in a sequence. This combination achieved better performance when compared to shallow GRU based models.

3.2 Reconstruction Error based AD for Sequential data

The second approach to AD for sequential data is to design a robust and computationally efficient HIDS using an Encoder-Decoder mechanism. The work in this section outlines the work from the second paper titled *"Bidirectional LSTM Autoencoder for Sequence based Anomaly Detection in Cyber Security"* (Chawla et al. 2019). The bidirectional encoder learns a compressed vector representation of input data, and the decoder regenerates the input data from this compressed representation. The target sequence reconstructs the normal behavior, and thereafter uses the reconstructed sequences to detect the anomalies. The concept of autoencoders have been considered as a possible approach to extend interpretability for neural network based AD solutions.

The reason a second approach has been implemented for the the problem of anomaly detection in Cyber Security is that, Chawla et al. (2020) have shown that autoencoders can be made interpretable. Interpretability is examined in chapter 4.

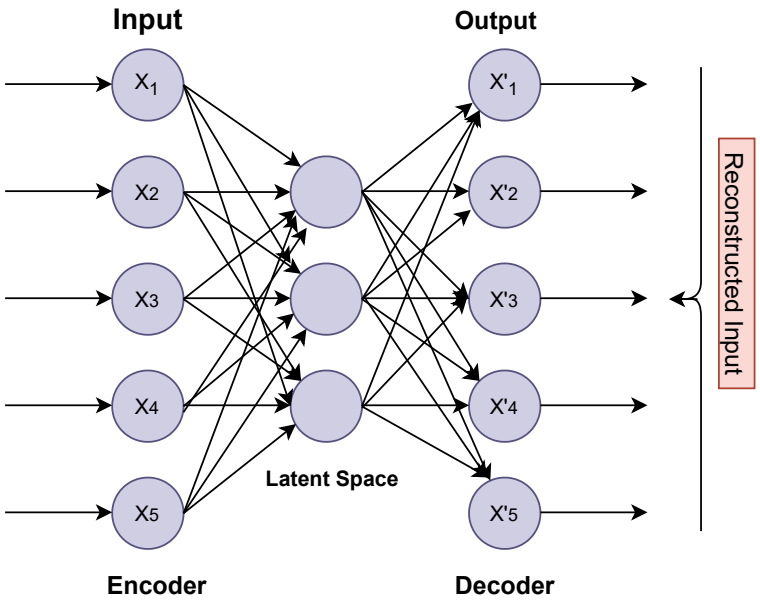


Figure 3.5: Autoencoder

An autoencoder, an unsupervised learning technique, is a neural network that is trained to produce an output which is the same as its input. Since the concept of autoencoders does not have dedicated targets, hence it is trained in an unsupervised manner to learn data representations. An autoencoder involves dimensionality reduction of high dimensional data, like Principal Component Analysis (PCA) (Abdi & Williams 2010). The difference lies in the fact that autoencoders have shown to be more efficient in case of nonlinear encoding/decoding compared to PCA (Alkhayrat et al. 2020).

3.2.1 Architectural Design & Methodology

Fig. 3.6 shows a Bidirectional Autoencoder based model. The input to the model is a call sequence, which vary from 1 to 340, transformed into a dense vector of size 64. The Keras performs word embedding and transforms one hot encoding of integers in the call sequence. Embedding layer weights are learned during training, that is a pre-trained embedding is not used. The Batch Normalization layers were incorporated to further reduce the internal covariate shift, by normalizing layer inputs, resulting in faster training. The Batch Normalization layer helps with gradient propagation and significantly increases the training speed.

The Bidirectional Autoencoder is built using LSTM units for the decoder, which has the same number of layers and units as the encoder. The bidirectional encoder and unidirectional decoder part of the model are jointly trained. The encoder part of the model reads the variable length input and compresses the input data into a latent space representation, known as a context vector. The encoder units are stateful and the output from the encoders is discarded. The model is trained using Adam optimizer with a mini batch size of 64, with categorical crossentropy as the loss function (Kingma & Ba 2015). The model outputs the probability distribution for the next system call using the softmax activation function as the output of the dense layer.

During the inference process, the encoder is initiated using the encoder inputs from the training along with the encoder states. The decoder is then initialized by using decoder states inputs (hidden state + cell state) along with the shared embeddings. The decoder layer returns both state and sequences to generate the target sequence probabilities.

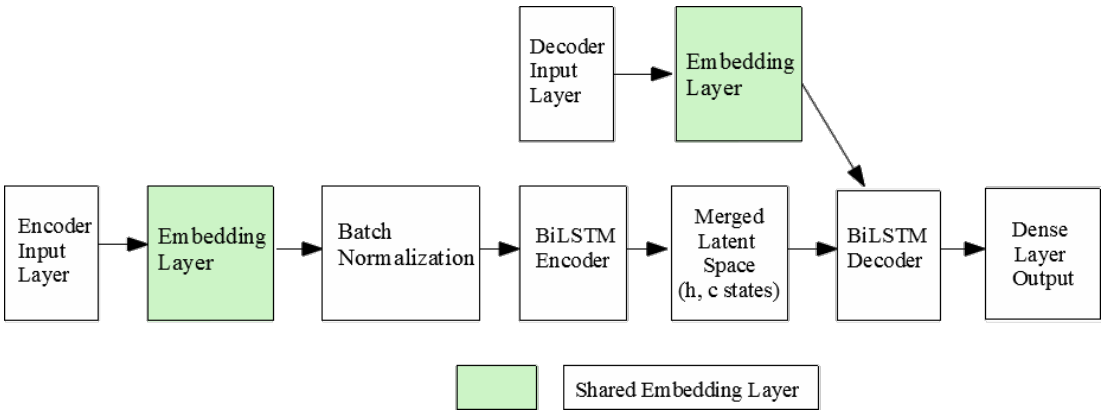


Figure 3.6: Architecture for predicting Anomalous Sequences

3.2.2 Probability using Reconstruction Error

The model estimates the probability of a sequence occurring using the conditional probability distributions. The output layer is a Dense layer, essentially a regular densely connected neural network layer. It is used with a softmax activation function in order to predict a probability distribution for the next integer in the call sequence.

3.2.3 Training the Sequences using Bidirectional RNN

The data set was split into two partitions, training and testing set. A Bidirectional CuDNNLSTM based encoder decoder was trained with two third of normal system call traces (normal + validation) 3470 traces. The rest one third of normal system call traces (1735) and attack sequences (746), total 2481 system call traces are tested after the model is trained.

The specification of the computational machine includes Intel core i7-8700@3.20GHz processor, 16GB of RAM and NVIDIA GeForce GTX1070 GPU running 64-bit Windows 10 operating system and the NVIDIA CUDA Deep Neural Network library (CuDNN). The Keras python library was used running on top of a source build of TensorFlow 1.7.1 with CUDA support (François Chollet 2022).

The model as shown in Fig. 3.7 with 200 units gave the best value (0.86) for the Area Under the ROC curve (AUC). CuDNNLSTM with 170 was behind resulting in an AUC value of 0.81. The model produces approximately 90% True Detection Rate with a False Alarm Rate of 25%.

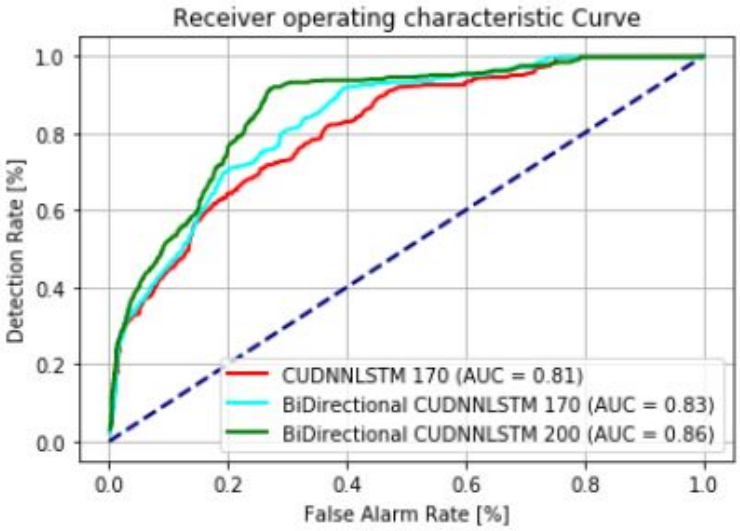


Figure 3.7: ROC curve comparing different models using Bidirectional Autoencoder

Note: CuDNNLSTM are variants of LSTM which are supported only on GPU systems with the Tensorflow-gpu backend. CuDNNLSTMs have proved to significantly faster as compared to LSTM (Zhang et al. 2018). By default, CuDNNLSTM supports tanh activation function.

3.2.4 Discussion on Reconstruction Error Approach for AD

In this experiment, RNN based Encoder-Decoder model was proposed, where the entire sequence is read by the neural network. The experiment evaluated a bidirectional CuDNNLSTM which can read the information from both the past and the future. It was observed that the CuDNNLSTM language model implementation has substantially reduced the training time when compared to an LSTM model. The use of large mini-batches as an input to the model increases the available computational parallelism, thereby reducing the overall training time. The time taken for CuDNNLSTM is approximately 10x faster than LSTM due to faster convergence in training.

Secondly, the area under the curve as compared to the previous model, Chawla et al. (2018) significantly improved AUC score of 0.86. There was no specific requirement to reverse the source/target sequences, as mentioned in most of the encoder decoder papers for effective Encoder-Decoder learning. Sutskever et al. (2014) reversed the order of words to "**establish communication**" between the input and the output to boost the performance of the LSTM. for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM

is mapped to c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c. This way, a is in close proximity to α, b is fairly close to, β. The proposed bidirectional autoencoder leverages shared embeddings between the encoder and the decoder of the model. In shared embeddings, parameter tying reduces the number of parameters that need to be learned by the model. With this approach, a single end-to-end model can be trained directly on source and target sequences without reversing the words, thereby making computation more efficient. The empirical evaluation demonstrates that this experiment has improved the overall anomaly detection performance, in terms of AUC score.

The proposed solution has maintained near state-of-the-art performance by designing computationally efficient models, with a substantial smaller training times when compared to the other traditional machine and deep learning models. The novel design of incorporating shared embedding layer with the merged latent space achieved better AUC of 0.86 with and accuracy of 90% and False Alarm Rate of 25%.

This approach introduces the efficient use of an autoencoder based reconstruction error concept, which can then be used for interpretability to perform troubleshooting and aid in root cause analysis. This research will address network performance management based KPI vulnerabilities in the telecommunications industry through novel architectures,discussed in chapter 4. In addition, the implementation of DNNs with an 'interpretable mechanism' will in effect change what is normally a black box technique into a white box technique enabling location of contributing factors causing anomalies in the network. In the framework, the top features with high reconstruction errors are connected to the features that contribute most to reconstruction error. The unsupervised approach assists the analyst in identifying anomalies, categorizing them, and identifying their causes. This greatly speeds up the root cause analysis process. Chapter 4 aims to address the broader horizon of discovering the unknown events by developing efficient, interpretable and fast neural network-based solutions.

3.3 Summary

In conclusion, the answer to research question no 1 is that prediction based and reconstruction error based AD can effectively detect the anomalies in sequential data. The proposed combined convolutional neural network (CNN) and recurrent neural network (RNN) model as an anomaly detection system using the ADFA-LD dataset. The CNN layers capture the local correlations of structures in the system call traces and RNN layer learns the sequential correlations from the features. This model is trained on normal traces and predicted the probability distribution for the next system call in each trace. Accordingly, it predicts the probability for the entire trace, and imposing a threshold based on the range of the negative log likelihood, it classifies new traces into normal or abnormal class.

In the employed reconstruction error based Bidirectional LSTM, the data is first fed into the input layer in the form of word embeddings. The embeddings are then propagated to a second layer called the hidden or bottleneck layer (where learning is stored) which are later then reconstructed at the output layer. A bidirectional RNN exploits the idea to improve on the performance of chronological-order RNNs. It looks at its input sequence both ways, obtaining potentially richer representations and capturing patterns that may have been missed by the chronological-order version (François Chollet 2018). The model produces 90% accuracy and a 25% false alarm rate.

The algorithms designed and implemented are computationally inexpensive in comparison to the traditional deep neural network models. Their accuracy is comparable to that of traditional AD methods, with reduced training and testing times. The empirical evaluation demonstrates that this experiment has improved the overall anomaly detection performance, in terms of AUC score.

More research is needed to discover the interpretations in sequential AD. In order to identify and investigate, chapter 4 will present how autoencoders mechanism can be extended further to neural network interpretability. The use of the reduced representation in autoencoders will be a great benefit to anomaly detection applications.

Chapter 4

Interpretable Anomaly Detection

This chapter outlines two approaches to anomaly detection for network data in telecommunication. The approach outlined in the first subsection (4.1) is an approach applied on a RAN data-set with the global and local interpretations. The second subsection (4.2) is an extension on the first approach which outlines an interpretable anomaly detection solution using an internal feedback loop.

***Research Question 2**: Is it possible to design a framework for Deep Neural Network Anomaly Detection, with enhanced interpretability and performance, to support telecommunications network troubleshooting and root cause analysis?*

The goal of this chapter is to define a reference architecture for end-to-end interpretable feedback-guided anomaly detection solution. The approach adopts an interpretable framework for categorization of anomalies that helps analysts to find most relevant anomalies faster. The techniques are applied to address the problem of false positives which can be a problem with anomaly detection solutions (Patcha & Park 2007). These experiments will answer the second research question.

The sophistication of Deep Neural Networks (DNNs) algorithms has recently increased with the ability to process massive network management data sets while extracting useful features to quickly identify anomalies (Fadlullah et al. 2017, Abdullah Al Mamun & Valimaki 2018, Fernandez Maimo et al. 2018). These algorithms are often difficult to interpret and their "*black box*" nature makes it difficult for stakeholders to understand the model's output. Interpretable explanations for anomalies is a relatively new addition to the study of anomaly detection.

The thesis focus on the importance of interpretation of anomalies detected by DNNs and a solution providing explanations for experts to better understand the reason behind decisions made by a model. Anomaly detection methods in general not only detects signature based anomalies but also anomalies which could be missed by experts who rely on domain rules. The architecture take the findings one step further and incorporate an internal automated feedback loop into an interpretable anomaly detection framework. The data used for the experiments is Ericsson proprietary Performance Management (PM) data set which is referred to as network cell trace data in this section. The proposed algorithm shows how an internal feedback loop incorporated into an unsupervised training mechanism improves feature learning by removing irrelevant or redundant features, which are normally known only to domain experts.

4.1 IUAD Framework: Interpretable Unsupervised Anomaly Detection

A lot of effort is spent doing analysis of cell trace files to understand why network fails and what is the reason for the bad network. In such scenarios, DNNs have shown impressive results and state-of-the-art performances in the field of telecommunications (Abdillah et al. 2016, Mnih et al. 2016). Multivariate AD has become one of the most challenging aspects of modern digital technology and it has become imperative to minimize and possibly avoid the impact of faults in network performance management.

The IUAD framework employs autoencoders to detect anomalies and SHAP to identify contributing anomalous features, which can be interpreted by the domain experts. Third paper titled, *"Interpretable Unsupervised Anomaly Detection For RAN Cell Trace Analysis"* demonstrate the effectiveness of the proposal whereby providing human interpretable multi-feature anomaly detection (Chawla et al. 2020). A key task is the classification and identification of anomalous operation modes (and faults). This is important to separate them from normal operation conditions. In addition, these diagnoses should be interpretable by domain experts to:

1. Gain acceptance by these experts.
2. Support effective root cause analysis and localisation. The analysis of multivariate network data in order to identify anomalous data instances. Root cause analysis benefits from this by filtering features whose values lead (to some extent) to such anomalies.

The IUAD framework focuses on the importance of interpretation of AI models and offers an explainable solution. The solution provides stakeholder experts to better understand the reason behind decisions made by model and hence aid in faster troubleshooting.

4.1.1 Architecture & Methodology

Fig. 4.1 illustrates an unsupervised anomaly detection framework which detects anomalous behaviour on the network. The framework assists Ericsson Engineers for interpreting network insights for telecom data sets. It aims to answer key questions about anomalous attack data patterns which remain hidden in the unstructured raw data. The data source of the Data Stream Processing will be discussed in section 4.1.2.1.

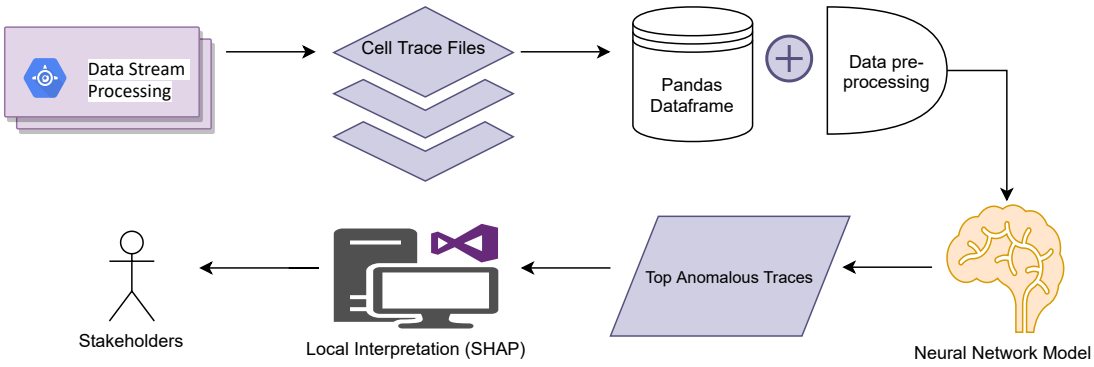


Figure 4.1: IUAD Framework for Anomaly Detection

4.1.1.1 Data Wrangling

Data preparation is often considered the main time consuming step of any machine learning task. Data wrangling pipeline plays an important role to facilitate the training/testing process by appropriately transforming and scaling the entire data set. The following are some of data pre-processing steps carried out:

(a) **Drop features with zero values or consistent across**: This data cleaning step removes the features which has either zero or consistent (same) value across the data set. Additionally, the features were removed which has negative values for most of the time in the data set and don't contribute in any feature learning. A negative KPI values occur when data is not available at the eNodeB site (section 4.1.2.1) .

(b) **Check Null values:** Removing null values from the data set is one of the important steps in data pre-processing. These null values adversely affect the performance and accuracy of any machine learning algorithm.

(c) **Log transformation:** Since the data set consists of variables of several magnitudes, log transformation has been applied to the entire data set. This helps in reducing the variability and skewness of data which leads to better accuracy of the model. The log transformation is obtained by computing:

$$f(x) = np.log10(x + 1 - min(x)) \tag{4.1}$$

(d) **Feature normalization:**This is done to ensure that the entire data set is scaled to [0,1]. Additionally it helps in faster convergence of the gradient-based neural network learning process.

4.1.1.2 Autoencoder Model

Autoencoders (see Chapter 2) learns a compressed representation of the input data and reconstructs the input from this representation. Sparse Autoencoders impose a sparsity constraint (limiting the number of active neurons participating in processing inputs) where they learn feature representations in the data even for a large number of hidden units (*CS294A Lecture notes, Sparse autoencoder* 2011). Additionally, L1 norm regularization learns sparse and enlarged representation of the input data (Ng 2004). In Keras, this can be done by adding an activity regularizer to the Dense layer. In the proposed algorithm, sparse autoencoder has been implemented with L1 regularization of 3e-7 with MSE loss which learnt better representation than vanilla autoencoders.

4.1.1.3 SHAP Interpretation

The KernelExplainer function (refer to Algorithm 1) takes as input the model, an instance of anomalous input x and a set of background instances. For a particular feature x_i' it calculates a set of SHAP values which measures the importance of each of the features $x_1, x_2 .... x_n$ in predicting x_i'. Here x_i' is the reconstructed input feature from the trained model. The local interpretability can be represented graphically by using force plots which enables us to pinpoint the SHAP value of features with respect to each other.

4.1.1.4 Interpretation

Algorithm 1 explains the process how SHapley values are being interpreted to explain the cell trace anomalies using SHAP output. The Kernel Explainer SHapley function outputs allFeatures, shapValue and inputValue for an anomalous cell trace. The shapValue corresponds to the impact of each feature on all other features in the cell trace. The features are sorted based on the shapValue in descending order. Then, the positive shapValue features and their corresponding inputValue are stored in the CSV output as Effect (*featureEffect*) and Value (*featureValue*) respectively. Lastly, it displays the top unique contributing feature.

Algorithm 1 Autoencoder Anomalies Interpretation

Input: trainData - Background instances
f - Autoencoder model
x - Anomalous cell trace to be explained
Output: SHAP values saved in CSV file and Top Contributors Feature list for stakeholders.

```
1:  procedure ANOMALIESINTERPRETATION
2:      allFeatures, shapValue, inputValue ← KernelExplainer.getshapValues(f, trainData, x)
3:      sortedFeatures ← sortDecrease[allFeatures, shapValue]
4:      for i in sortedFeatures.length: do
5:          feature ← sortedFeatures[i]
6:          featureEffect ← feature.shapValue
7:          featureValue ← feature.inputValue
8:          if featureEffect > 0 then
9:              print2CSV(featureValue, featureEffect)
10:         end if
11:         topContributor ← sortedFeatures[0]
12:     end for
13:     print Unique topContributor
14: end procedure
```

4.1.2 Experimental Results

All experiments have been conducted on the computational machine which includes Intel® Core™ i7-8650U CPU @1.90GHz × 8, 25.8 GiB memory, Ubuntu 18.04.4 LTS operating system. The Keras python library 2.3.1 was used for running on top of a source build of Tensorflow 2.1.0.

4.1.2.1 Radio Access Network Cell Trace data set

The cell trace data for this study is collected from multiple Evolved NodeB (eNodeBs) as csv files. There are a mix of periodic and procedural events in each file. Periodic events are produced at fixed intervals capturing key metrics. Procedural events are produced every time the node or User Equipment(UE) performs an operation e.g. UE Context Setup, Handover Preparation, etc. Fig 4.2 illustrates the data stream solution architecture which collects events from eNodeB. These event streams are parsed by the events parser and sent upstream to the correlation engine to produce a coherent data set. This data set is collection of session records which includes UE, eNodeB and EPC.

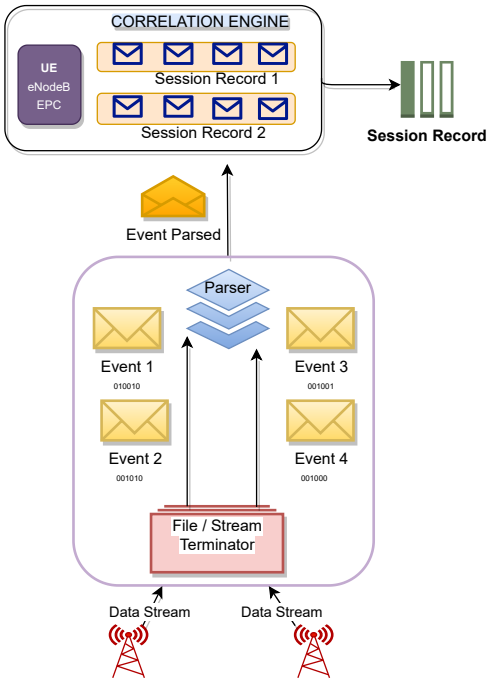


Figure 4.2: Data Stream Processing

The output from the above process produced cell trace files which provided the input data for anomaly detection using an autoencoder and subsequent interpretation using correlation and SHAP values.

The data worked upon is a Performance Management (PM) data set (Ericsson Propriety) which is referred to as network cell trace data in Table 4.1. The table shows the cell trace data related event values that which are produced at fixed regular intervals for a number of different resources i.e. per cell, per radio, per UE Bearer and per UE Traffic Reports. The columns or features are outputted by the correlation engine for each session. The following table shows the sample records:

Table 4.1: Session Record Sample Output

Index	Date	StartCell Id	ENB S1 APID	MME S1 APID	RAC_UE_REF	S1 Release Cause	Vol Total Downlink	Last RSRP	..
0	36:49:9	24	271760	143110830	290586748	8	831	26	..
1	36:04:6	26	27138	59878751	290455681	8	447	33	..
2	36:50:9	34	271863	92589077	290586754	4	3927	31	..
3	36:39:1	37	278956	92589079	290455678	26	4589	20	..
4	36:50:1	37	271764	92589089	290455696	10	3947	14	..
...	..	..	..	..	..	..	..	..	..
36809	38:02:6	35	271789	92589096	290455612	1	3978	27	..
36810	37:52:1	36	271746	992589042	290586745	8	3912	14	..

Here, Table 4.1 shows the session record sample output collected for different UEs across cells. Mobility Management Entity (MME) S1 APID denotes eNodeB during first S1 Application Protocol (S1AP) message. The S1-MME interface connects the MME to eNodeBs and to UEs through the eNodeB, where S1AP procedures provide the signaling service between the eNodeB and the MME.

When a UE connects to an eNB, the eNB will assign a temporary identification number to the UE called Radio Admission Control User Equipment Reference (RAC_UE_REF). The UE has this RAC_UE_REF as long as it is actively connected to the eNB. If the UE becomes inactive or if there is a successful handover this RAC_UE_REF will no longer be in use.

S1 Release Cause code details if the session ended for a normal or abnormal reason. These are defined by the network experts based on telecom domain policy and rules. Vol Total downlink represents the data volume in downlink mode (Bonald & Proutiere 2003). Last RSRP denotes Reference Signal Received Power. It is the power of the LTE Reference Signals spread over the full bandwidth. A minimum of -20 dB Signal to Interference & Noise Ratio (SINR) is needed to detect RSRP.

Likewise, the session record output has total 100 columns, which will be used for training the model in section 4.1.2.2.

Note: The data set includes one of the cheat column, called as "S1 Release Cause" which details if the session ended for a normal or abnormal reason. For the purposes of training, this column has not been used by the proposed autoencoder model. This column has been used later for sanity check of the model.

4.1.2.2 Training the Model

The sparse autoencoder model architecture comprises of an input layer, 3 hidden layers and finally an output layer. Inspired by *Building Autoencoders in Keras* (2016), four different models were designed.

- 100-50-100 neuron units
- 300-150-300 neuron units
- 400-20-400 neuron units
- 500-250-500 neuron units

The model design matched the cardinality of the input and output layer. The experiments comprised of the above 4 deep stacked perceptron models to train approximately 2 million rows by tweaking the hyperparameters. The model is trained in minibatches (batch of 32 inputs at a time) gradient descent with the set of normal cell traces correlated data set. The model optimizes the Mean Squared Error (MSE) loss using Adam optimizer with a learning rate of 0.0008, followed by non linear activation relu function (used in training with dense layer defined), defined as:

$$f(x) = max(0, x) \tag{4.2}$$

The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

Callback with an early stopping mechanism of validation loss has been used as one of the regularization technique to avoid model overfitting. This technique stops training at the point when performance on a validation data set stops improving. The model trained after tweaking the hyperparameters is then used to reconstruct the input data. The reconstruction errors measures the distance between the original input and its autoencoder reconstruction (Shin & Kim 2020). The reconstruction errors calculated are then used to detect the anomalies in the test set correlated cell trace file.

4.1.2.3 Identifying the Anomalies

As the MSE was to be used as the key metric error indicator, average RMSE for normal and anomalous instances were obtained. RMSE for normal instances and anomalous instances are 0.0197 and 0.2215 respectively. This seems to suggest that RMSE could be used to identify anomalous instances. Then, True Positive and False Positive rates are calculated for different threshold values to find anomalous cell trace instances. The AUROC (Area Under Receiver Operating Characteristic) and AUPR (Area Under Precision-Recall) are holistic metrics that summarize the performance of a detection method across multiple thresholds (Saito & Rehmsmeier 2015). Table 4.2 shows performance values for several model configurations. The best-fitted model (lowest marginal error) with 500-250-500 neuron units produces 0.96 AUC with 0.70 AUPR.

Table 4.2: Performance Values for Models

Parameters	Neurons	Learning Rate	AUROC	AUPR
46,631	100-50-100	0.0008	0.88	0.50
199,531	300-150-300	0.0008	0.95	0.68
305,981	400-200-400	0.0008	0.94	0.60
432,431	500-250-500	0.0008	**0.96**	**0.70**

4.1.2.4 Global Interpretation - Correlation Coefficients

Global interpretations (see Chapter 2) approach helps to understand the distribution of the target outcome based on the features. This level of interpretability is about understanding how the model makes decisions, based on a holistic view of its features and each of the learned components such as weights, other parameters, and structures.

Here, Fig. 4.3 shows global interpretation results i.e. strongly correlated features in the top anomalous rows detected by the trained autoencoder model. For the purposes of evaluation, anomalous rows were sorted based on the root mean squared error obtained and top 100 anomalous cell traces were obtained for further analysis. To get a representation of the relationships between features for these top anomalous cell traces, the pairwise correlation coefficients between features were calculated (among 100 columns discussed in section 4.1.2.1).

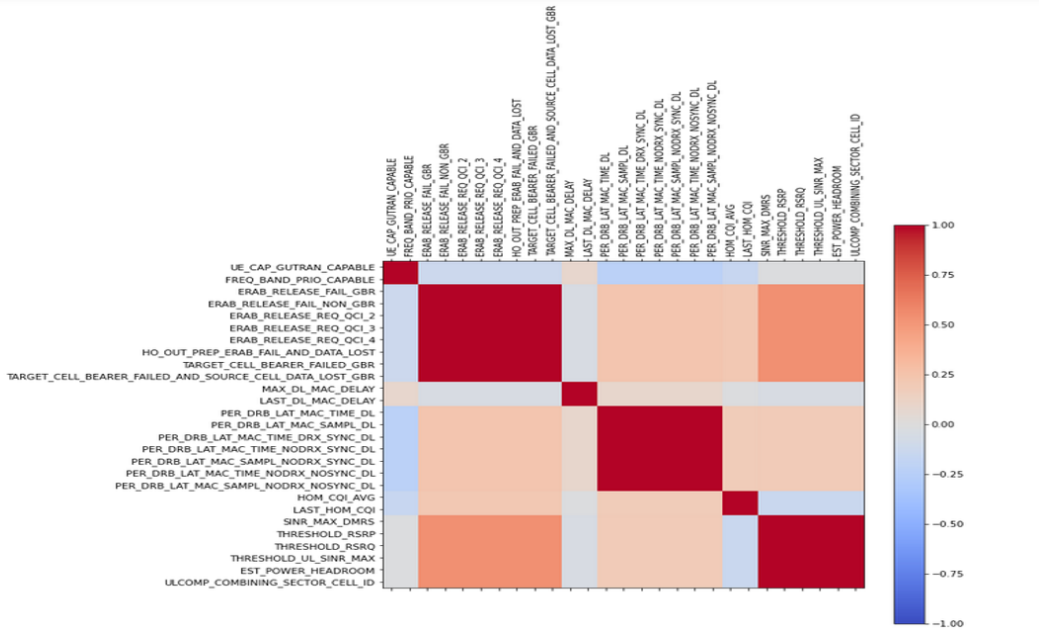


Figure 4.3: Collection of Anomalous Features

Fig. 4.3 highlights the snippet of collection of feature attributes in the top 100 anomalous cell traces. Higher the correlation score, depicts the stronger association among features. This helps the network engineers to identify the important set of correlated features.

4.1.2.5 Local Interpretation - SHAP values

Local interpretations (see Chapter 2) focuses on specifics of each individual and provides explanations that can lead to a better understanding of the feature contribution in individuals and smaller groups that are often overlooked by the global interpretation techniques.

In this section, the local interpretation results generated by SHAP has been reviwed. The KernelSHAP Explainer as described in Chapter 2 outputs features, SHAP values and input value for anomalous rows. Table 4.7 illustrates the contribution of input features to a sample anomalous row where features with higher Effect value indicate higher impact on the entry being anomalous.

One of the feature column was added to perform sanity check if the model results hold true or not. The row highlighted in Table 4.4 red shows the S1 Release cause 26 as anomalous cell trace session. When validated against the ground truth labels, it turned out to be a true positive. This particular entry indicates that a bad cell trace happening due to handover failure could be caused by no available neighboring Reference Signal Received Power (RSRP) (**Feature 109**), E-UTRAN Radio Access Bearer (ERAB) fail and data lost (**Feature 108**), high latency time (**Feature 75**) and low aggregated time for downlink delay in Media Access Control Address (MAC) layer (**Feature 173**).

UE_CTXT_RELEASE_CAUSE_S1	LAST_RSRP	LAST_RSRQ	THRESHOLD_RSRQ	reconstruction_error
0	-1	-1	-1	0.015394671
0	-1	-1	-1	0.014602758
11	-1	-1	-1	0.009074979
11	21	18	32768	0.008587354
9	-1	-1	-1	0.007776617
19	18	9	32768	0.007623958
9	-1	-1	-1	0.007387069
26	12	15	32768	0.007003561
11	-1	-1	-1	0.006574195
9	24	7	32768	0.006454201
9	-1	-1	-1	0.006099894
26	16	3	32768	0.00547743
26	15	10	32768	0.005472872
11	-1	-1	-1	0.00537023

Figure 4.4: Top anomalous rows detected sorted by reconstruction error

Similar to this example, the SHAP values can help explain the impact of features to each anomalous entry detected by the proposed autoencoder model.

The proposed solution prediction and the confirmation by the network engineers validated that ERAB fail and data lost was the most important feature for failed handover, which contributed for bad RSRP, increased latency and low aggregated downlink delay.

Table 4.3: Kernel Explanation- Contributing Features

Feature 108	Feature 109	Feature 75	Feature 173
Value = 0	Value = -1	Value = 2147483648	Value = 806
Effect=**0.050**	Effect=0.0054	Effect=0.016	Effect=0.0014

4.1.3 Discussion on IUAD Framework

Most state-of-the-art techniques tackle the problem of detecting network anomalies with high precision but the models don't provide an interpretable solution. This makes it hard for operators to adopt the given solutions. Reliable explanations help in interpreting anomalies, builds trust with users, helps identify contributing factors and removes barriers to entry for the deployment of deep neural networks in different domains. The design of a neural network model and associated interpretation techniques was proposed with a view to provide insights and support for the troubleshooting of network cell trace data without prior domain knowledge. This automated solution can save a lot of manual troubleshooting, with a resulting reduction in costs and resources.

The experimental results have been able to show that sparse autoencoders can be trained to reconstruct the input data and reconstruction error can be used to identify previously unseen anomalies. The empirical results achieved an AUC score of 0.96. One possible threshold for anomaly detection gives us a true positive rate of 0.9 and a false positive rate of 0.1. The AUPR curve was 0.70. This research has been carried out on Ericsson real network performance management datasets, which is their proprietary information and cannot be released in public domain. All released research results carefully avoid mention of identifiable customers or identifiable network users. While the work has been implemented and evaluated using prototype Ericsson's datasets and testing environments, the results has been demonstrated using anonymous data.

Global interpretability of a set of anomalous data is supported using pairwise correlations of features as stated in section 4.1.2.4. This level of interpretability assists Ericsson Engineers in understanding the distribution of the target outcome based on the features. A pairwise correlation was shown as a representation of the relationship between the top anomalous cell traces.

The local explanations provide a deeper insight for every anomalous trace. The SHAP values identify the features and the extent to which they have caused the anomaly. The kernel SHAP values help to explain anomalies detected by the proposed IUAD framework, which is an unsupervised model. As compared to the proposed approaches, Goodall et al. (2018) presented a streaming anomaly detection and visualisation system for discovering and explaining anomalous behavior in computer network traffic and logs. In addition to investigating the most anomalous events and IP addresses, the authors provided context to assist operators understand why they are anomalous.

The proposed model-agnostic method of explaining anomalous instances aims to provide a comprehensive explanation to the experts by focusing on the connection between the features with high reconstruction error and the features that are most important in affecting the reconstruction error (section 2.4.1.2). The output of such algorithms may identify anomalous instances that the domain expert was previously unaware of. An explanation of why an instance is anomalous can increase the domain expert's trust in the algorithm. The experiments on a real-world telecom network data set demonstrate the efficacy of the proposed algorithm. The framework predicted the contributing features were indeed responsible for the anomalous cell traces. The confirmation of this by the network engineers validated the proposed solution.

4.2 The Cluster Characterized Autoencoder (CCA)

The major contribution of this thesis was the development of a novel framework for detecting anomalies using subspace clustering and interpretability of candidate anomalies. The paper titled, *"Towards Interpretable Anomaly Detection: Unsupervised Deep Neural Network Approach using Feedback Loop"*, NOMS 2022, proposed a "Cluster Characterized Autoencoder" (CCA) based architecture which improves the model interpretability by designing an end-to-end data driven AI-based framework (Chawla et al. 2022).

The design, integration, and demonstration evaluates an unsupervised approach based on an autoencoder anomaly detection mechanism and apply it to network cell trace data. The interpretability results are achieved by validating the true positives/false positives of a re-trained model as the set of less important network attributes are removed. This multi-anomaly functionality allows for the identification of multiple anomalous network attributes. This approach extends the interpretability and effectiveness of a neural network based unsupervised anomaly detection solution.

4.2.1 Architectural Design & Methodology

In this section, the design of a Cluster Characterized Autoencoder (CCA) framework has been discussed. Fig 4.5 below shows the architecture of the solution. To the best of the author's knowledge, this work will be the first to implement anomaly detection solutions using an internal automated feedback loop in an unsupervised manner. Note that the part of this experiment contribution enables the interpretability of data-driven solutions.

Greedy layer wise training is used to train an autoencoder model in a number of phases. Once the model is trained, the reconstruction error is used as a metric on which the entropy based feature ranking algorithm is applied and a reduced set of features is used to retrain the model. This internal feedback loop is executed until the model's marginal error stops improving. Once the final model is trained, the candidate anomalies are predicted. The candidate anomalies are then classified using subspace clustering. Then based on chosen clusters, the SHAP explainer algorithm provides interpretability of the results. The interpretability framework provides network engineers with actionable insights which enables a deeper investigation of influencing features.

4.2.1.1 Data Wrangling

The columns with zero, static values were dropped which do not contribute to the training of a neural network model. The missing raw network data instances are imputed using k-nearest neighbor (KNN) (Keerin et al. 2012). Additionally, the feature columns of varying scale were normalized first by applying the log transformation, followed by min-max scaler normalization.

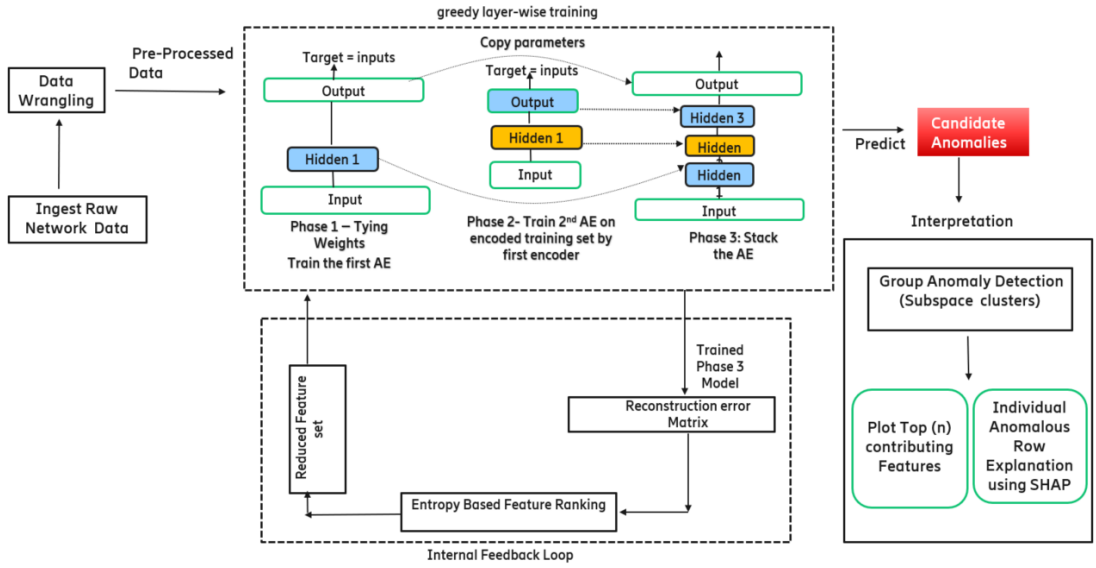


Figure 4.5: Cluster Characterized Autoencoder

4.2.1.2 Greedy layer wise training

Greedy layer wise training trains an autoencoder to reconstruct the original input data. The reconstruction errors calculated will be used to detect the anomalies in the test set network cell trace data. The Mean Squared Error (MSE) is used as the loss function for training the model and depends on the the difference between the actual (x) and reconstructed (x') attribute values. The Root Mean Squared Error (RMSE) is used as an evaluation metric to calculate the root mean squared error difference between the observed (x) and predicted (x') values of a set of n.

Greedy layer-wise training is used to train the autoencoder at each iteration of the feedback loop. Bengio et al. (2007) introduced the concept of greedy layer-wise training to train Deep Neural Networks (DNN). The basic idea of the greedy layer-wise training algorithm is to mitigate the difficult optimization problem of deep networks by better initializing the weights

of all the layers. So, instead of training the whole stacked autoencoder in one go, firstly the shallow autoencoder was trained to reconstruct the inputs (**Phase 1**). During the first phase of training, the first autoencoder learns to reconstruct the inputs. Then the whole training set was encoded using the first autoencoder and this gives us a new (compressed) training set. In **Phase 1**, the model tie the weights of the decoder layers to the weights of the encoder layers training by sharing the parameters. This results in faster training, by setting the decoder weights as transpose of the encoder weights. This limits the risk of overfitting and results in a lower reconstruction error than with a traditional neural network training methods (Erhan et al. 2010).

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(x_i - x_i^{'})^2} \tag{4.3}$$

Secondly, in **Phase 2**, the second autoencoder was trained on this new compressed data set. This way, the second autoencoder is trained on this new dataset. Finally, in **Phase 3**, a deep autoencoder was build and trained using the Phase 1 and Phase 2 autoencoders, i.e. firstly, the hidden layers of each autoencoder were stacked and then the output layers in reverse order.This gives the final stacked autoencoder. In this way the model was built and trained using a deep stacked autoencoder in an unsupervised manner using a greedy layer-wise approach. Inspired by Géron (2019), 3 training phases were chosen for the purposes of the experiment. Similar to this, very deep stacked autoencoders can be trained.

Here are the key benefits of this approach:

- Simplifies the training of deep stacked neural network model process.
- Facilitates the development of deeper networks.
- Lower generalization error.
- Useful as a weight initialization scheme.

4.2.1.3 Entropy-based Feedback Loop

While training a DNN model, not all the features contribute to its pattern learning. To further enhance the performance of the training, the concept of entropy based feature ranking have been implemented. Theoretically, the entropy is maximum when the reconstruction error are uniformly distributed in the feature space, whereas the entropy is low when the reconstruction errors are in well formed clusters.

For multidimensional data, the formula to calculate the distance between two instances is defined as:

$$Di_1, i_2 = \sqrt{\sum_{k=1}^{M} (\frac{x_{i_1 k} - x_{i_2 k}}{max_k - min_k})^2} \tag{4.4}$$

Algorithm 2 Feature Rank Algorithm

Input: Reconstruction error of features 1...M
Output: Reduced set of Features

```
procedure FEATURE RANKING
    distanceScore ← feature [1...M]
    entropyScore ← ConvertDistancetoEntropy
    sortedEntropyScore ← sortEntropyScore[Desc]
    topRankings ← 95%(sortEntropyScore)
end procedure
```

Here, x_{i_1} and x_{i_2} are the two reconstructed error data points obtained after training the model. The interval in the k^{th} dimension is normalized by dividing it by the maximum interval ($max_k - min_k$) before calculating the distance, where ($max_k - min_k$) are calculated using the reconstructed error matrix for features [1...M] (Dash & Liu 2000).

The distance between 2 data points in the feature space were calculated as the Euclidean distance as shown in Algorithm 2. Then, the distance to entropy between 2 data points are converted by calculating a similarity for numerical data using the following equation:

$$S_{i1,i2} = e^{-\alpha \times D_{i_1, i_2}} \tag{4.5}$$

of shape (n*n) where n is the number of data points 4.2.2.1. Entropy for two data points is defined as:

$$E = -\sum_{i1=1}^{N}\sum_{i2=1}^{N}(S_{i_1,i_2}) \times logS_{i_1,i_2} + (1 - S_{i_1,i_2}) \times log(1 - S_{i_1,i_2})) \tag{4.6}$$

where S_{i_1,i_2} values lies between 0 and 1. The similarity between 2 data points x_{i_1} and x_{i_2} is high if the 2 instances are very close and similarity is low if the 2 are far away. The features were sorted by entropy score and the top 95 percentile was used as feedback.

The feedback loop shown in Fig 4.6 forms an integral part of the end-to-end architecture in which the neural network is trained in an iterative manner. Using the feedback loop, autoencoders are given feedback on the importance of features by ranking and removing those with higher reconstruction errors. The reduced set of features thus helps in removing unwanted features, and thus improving the performance of the model which captures the "essence" of the data. This results in fewer resources to complete the training computations.

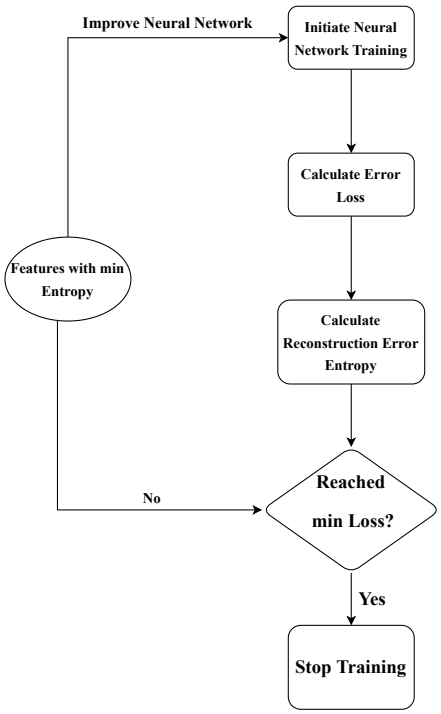


Figure 4.6: Feedback loop flowchart

Here, the internal feedback loop becomes a part of the deep learning model where the model trains until it reaches an equilibrium point (min loss margin). The reduced set of ranked features are passed through the model which speeds up model training, and allows the model to be less prone to test errors.

4.2.1.4 Separate TP/FP using Subspace Clustering

In order to investigate the ground-truth categories (true positives/ false positives), the clusters of candidate anomalies were identified using subspace clustering. Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces within a data set (L. et al. 2004). Related work on traditional clustering algorithms assumes a single pattern and/or does not look for subspaces (Rokach & Maimon 2005). Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, possibly overlapping subspaces. T-Distributed Stochastic Neighbor Embedding (t-SNE) representation visualizes the clusters (using subspace clustering) of anomalies.

Fig 4.7 shows t-SNE representation helps to identify a few, small micro-clusters of anomalies hidden in arbitrary feature subspaces. t-SNE visualizes high-dimensional data by giving each data point a location in a two or three-dimensional space (Van der Maaten & Hinton 2008). Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities.

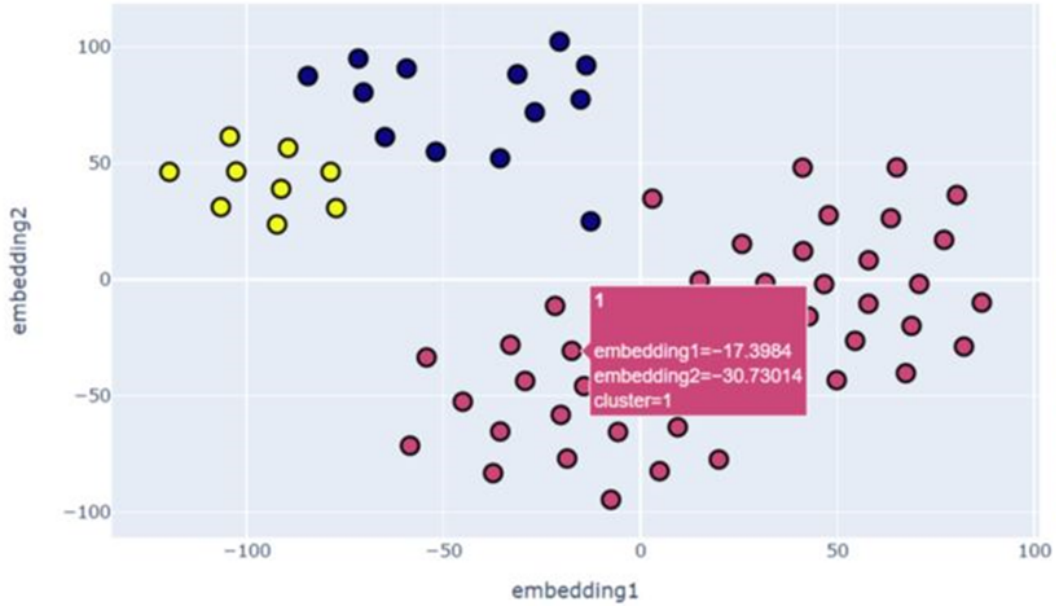


Figure 4.7: t-SNE representation of 3 clusters

t-SNE first computes probabilities p_{ij} that are proportional to the similarity of objects $\mathbf{x}_i$ and $\mathbf{x}_j$, given by:

$$p_{j|i} = \frac{exp(-||x_i - x_j^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k^2 / 2\sigma_i^2)} \tag{4.7}$$

where σ_i is the variance of the Gaussian centered on data point x_i.

4.2.1.5 Interpretability Framework

Fig. 4.8 shows the cross validation and interpretation of anomalies by the network engineer. The test samples (input data here) when tested against the trained model, predicts the candidate anomalies. These candidate anomalies are clustered using Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP).

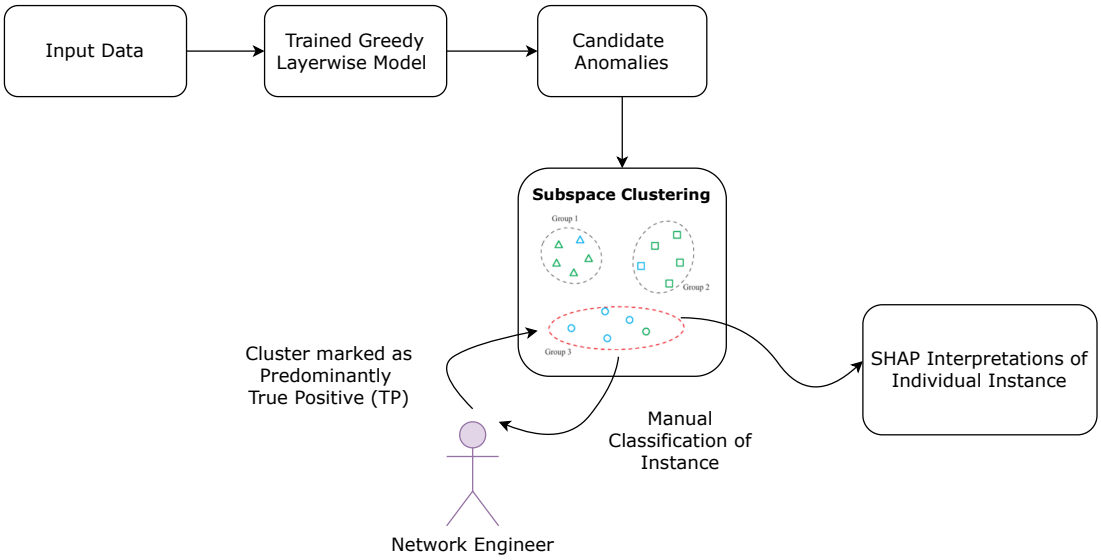


Figure 4.8: Interpretation of Anomalies

The subspace clustering when applied on the set of predicted candidate anomalies, identifies the set of clusters (using k-means clustering) which correspond closely to TP and FP. The network engineer picks one instance from a cluster to determine if the cluster contains mostly TP or FP. Then the engineer can choose any instance from the cluster and SHAP values for the instance are calculated.

The KernelExplainer SHAP function takes as input the model, an instance of anomalous input x and a set of background instances. For a particular feature x_i' it calculates a set of SHAP values which measures the importance of each of the features $x_1, x_2, ..., x_n$ in predicting x_i'. This local interpretability can be represented graphically by using force plots, decision plots which enable us to pinpoint the SHAP value of features with respect to each other. The complete implementation process is described in Algorithm 3.

Algorithm 3 Anomalous features Interpretation

Input: trainData - Background instances
x - shap.kmeans(trainData, 200)
model - trained Autoencoder
AnomalousRowsDF - Predicted anomalies based on reconstruction error
row - Anomalous row to be explained
Output: Top Anomalous Features

procedure ANOMALIESINTERPRETATION
    filterdf $\leftarrow AnomalousRowsDF[clusterID]$
    topFeatures $\leftarrow sortfilterdf[Desc, allFeatures]$
    topFeatures $\leftarrow topFeatures.Transpose$
    for feature in topFeatures.index: **do**
        weights, bias $\leftarrow model.getweights()$
        featureIndex $\leftarrow trainData.index[feature]$
        updatedWeights[featureIndex] $\leftarrow [0] * weights$
        model.setWeights $\leftarrow (updatedWeights, bias)$
        allFeatures, shapValue, inputValue $\leftarrow KernelExplainer(model.predict$,x)
        forcePlot $\leftarrow featureIndex, shapValue, row$
        decisionPlot $\leftarrow featureIndex, shapValue, row$
    end for
    print topFeatures
end procedure

procedure CONTRIBUTINGFEATURESINTERPRETATION
    allFeatures, shapValue, inputValue $\leftarrow forcePlot.getshapValues(row)$
    sortedFeatures $\leftarrow sort[Desc, allFeatures, featureEffect]$
    for feature in sortedFeatures.length: **do**
        feature $\leftarrow sortFeatures[feature]$
        featureEffect $\leftarrow feature.shapValue$
        featureValue $\leftarrow feature.inputValue$
        if $featureEffect > 0$ **then**
            print2CSV(featureValue, featureEffect)
        end if
    end for
end procedure

The pseudo code explains the process how SHapley values are being interpreted to explain the cell trace anomalies using SHAP output. In the anomalies interpretation procedure, first filter the anomalous rows based on the chosen cluster ID obtained using subspace clustering. Then for each anomalous row, first select the top features with the largest reconstruction error. Then, for each feature in the list of top features, set the model weights to [0] and use Kernel SHAP explainer to calculate the SHapley values.

The Kernel Explainer SHapley function outputs allFeatures, shapValue and inputValue for anomalous cell trace. The shapValue corresponds to the impact of each feature on all other features in the cell trace. The features are sorted based on the shapValue in descending order. Then, the positive shapValue features and their corresponding inputValue are stored as Effect (*featureEffect*) and Value (*featureValue*) respectively.

4.2.2 Experimental Results

4.2.2.1 Data Stream Processing

The network cell trace data set as shown in Fig. 4.9 is a collection of session records which includes UE, eNodeB and Evolved Packet Core. The output from this process produced

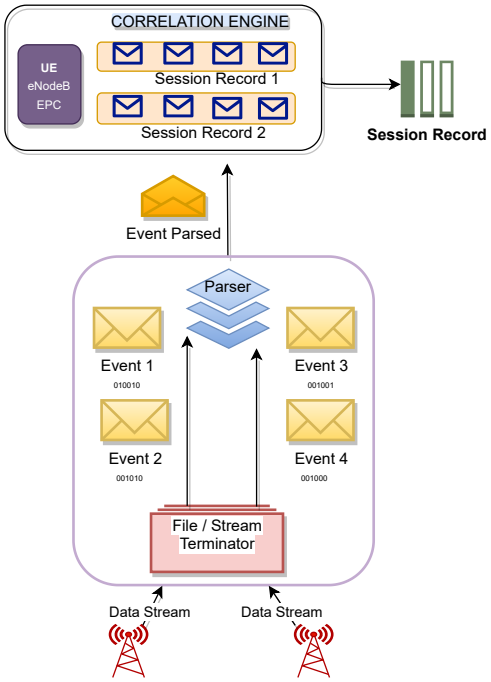


Figure 4.9: Data Stream Processing includes UE, eNodeB, EPC

cell trace files of 36,814 records (data points) and 539 network attributes (features) which provided the input data for anomaly detection using the referenced architecture providing an interpretable anomaly explainer.

4.2.2.2 Training

A sparse autoencoder model was trained using greedy layer-wise procedure (Section 4.2.1.2), where the model was trained one layer at a time, by fixing the parameters of the previous hidden layers, with unsupervised criterion. The architecture comprises of initializing the weights of all layers to improve generalization performance. Here, the 3 autoencoder models were trained in different phases with network dimensions set to 500-250-100 neuron units. The model design matched the cardinality of the input and output layer. Each layer is pre-trained in minibatches of size 32 with dropout rate of 10 - 20%. The final Deep Autoencoder (AE) model is further fine tuned for 30 epochs without dropout as shown in Fig 4.10.

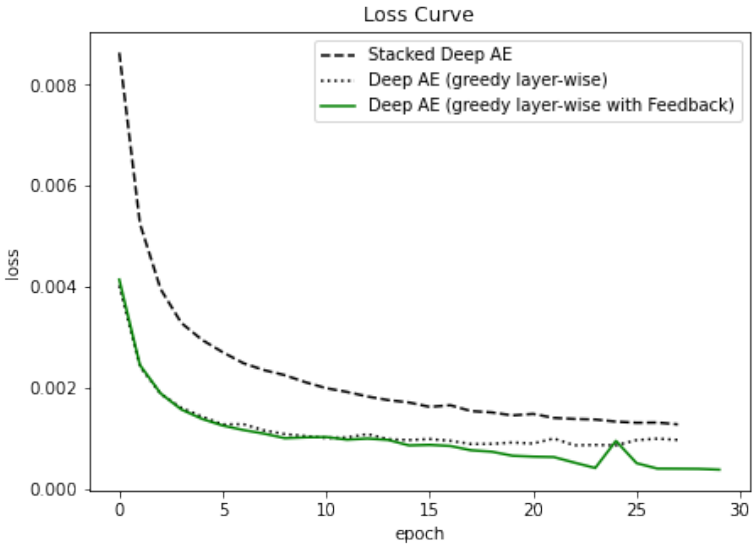


Figure 4.10: Error Distribution curve

The model optimizes the Mean Squared Error (MSE) loss using Adam optimizer, followed by the non linear activation function ReLU (Rectified Linear Unit) been used in training with each dense layer, defined as:

$$f(x) = max(0, x) \tag{4.8}$$

Table 4.4: Compare and evaluate different Models

Model	Train Error	Validation Error
Stacked Deep AE	0.0027	0.0029
Deep AE with greedy layer-wise	0.0011	0.0014
Deep AE with Feedback	0.0008	0.0004

Callback with an early stopping mechanism based on validation loss and dropout has been used as the regularization techniques to avoid model overfitting (François Chollet 2018). This technique stops training at the point when performance on a validation data set stops improving. The model trained after tweaking the hyperparameters is then used to reconstruct the input data.

4.2.2.3 Candidate Anomaly Detection

The RMSE for normal instances and anomalous instances are 0.129 and 0.3019 respectively. The mean absolute error (MAE) for normal test samples were **0.056637** and **0.10224** for anomalous instances. Fig 4.11 shows the MAE error distribution of the test samples. Here, the high RMSE and MAE scores depict the anomalous cell trace which were not reconstructed well by trained model.

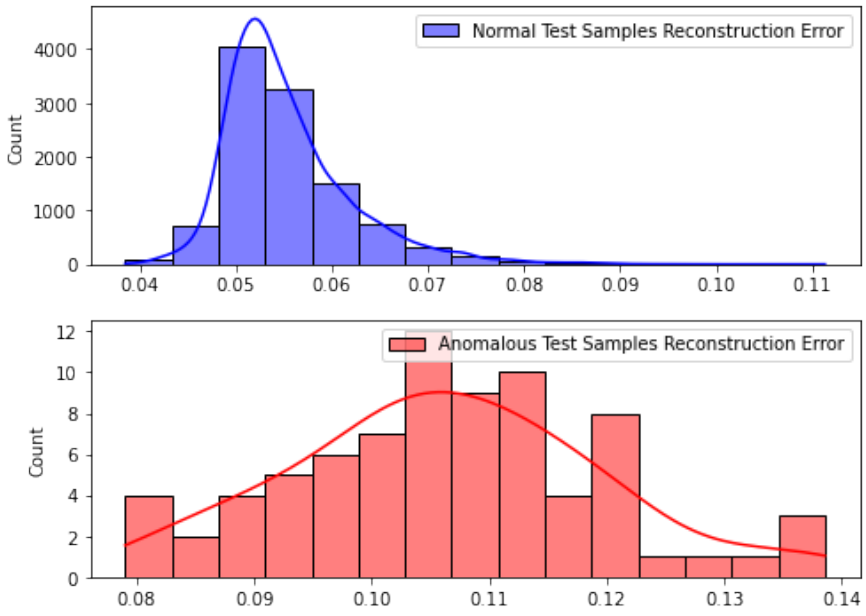


Figure 4.11: MAE Error Distribution

Empirically, the experimental results show that a greedy layer-wise trained autoencoder model trained using **normal** network data helps to optimize the modelling of normal data. The trained model then detects anomalies in multivariate network data. When given an anomalous instance, it may not be able to reconstruct it accurately, and this leads to higher reconstruction errors compared to the reconstruction errors for the normal cell traces. The test samples consists of 10968 normal samples and 77 anomalous cell traces.

The candidate anomalies are detected in the network cell trace data using an unsupervised technique where the labeled data is not normally available. Then, the outliers are detected in the test samples by calculating the robust Z-scores. It measures the outlier strength and is been considered more robust than normal Z-scores as it relies on the median for calculating the z score.

Robust Z-scores based on reconstruction error (MSE) are defined as:

$$Z_{score} = \frac{0.6745 \times (x_i - \tilde{x})}{MAD} \tag{4.9}$$

where 0.6475 is the 0.75th quartile of the standard normal distribution, x_i is the data points (MSE), $\tilde{x}$ is the median and MAD denotes the median absolute deviation. A Z-score greater than the threshold value (5) is determined as an anomaly (anomalous cell trace). The identified candidate anomalies (55) are then classified into clusters using subspace clustering.

Now for the data set been used, ground truth values exist indicating whether a trace instance should be anomalous or not. Note that this information is only used for evaluating the model.

True Positive and False Positive rates are calculated for different threshold values to find anomalous cell trace instances. The AUROC (Area Under Receiver Operating Characteristic) are holistic metrics that summarize the performance of a detection method across multiple thresholds (Bradley 1997). The best-fitted model (lowest marginal error) with 500-250-100 neuron units produces 0.9326 AUC as shown in Fig 4.12.

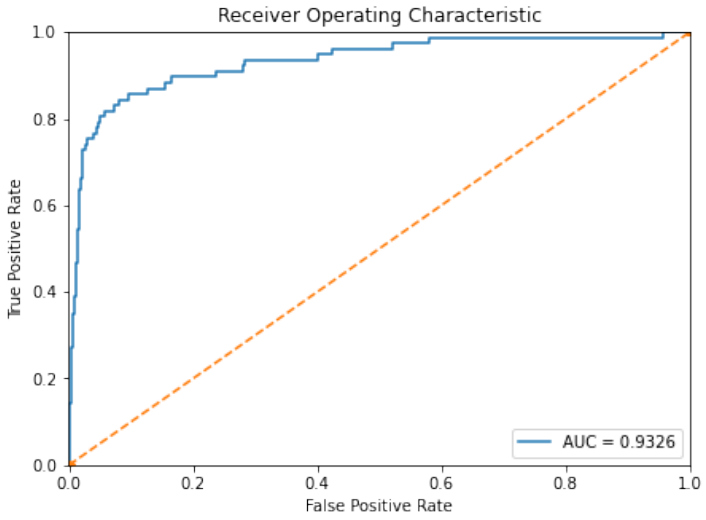


Figure 4.12: ROC with best fitted Feedback model

4.2.2.4 Clustering evaluation

The clustering evaluation shows how well the clustering of detected candidate anomalies was able to partition the true positives and false positives. To do this, the entropy technique is used in decision trees to measure the node impurity of nodes in the decision tree (Gulati et al. 2016). Here, the cluster impurity or how is measured to identify how well/badly the clusters separate true positive and false positives. Formally, the entropy of a cluster is defined as:

$$H(w) = -\sum_{c \in C} P(w_c) \log_2 P(w_c) \tag{4.10}$$

where c is a class in the set **C** of all classes and $P(w_c)$ is the probability of a data point being classified as c in cluster w. The total entropy of a clustering is defined as the weighted sum of these values:

$$H(\Omega) = \sum_{w \in \Omega} H(w) \frac{N_w}{N} \tag{4.11}$$

where $\Omega = w_1, w_2, \ldots, w_K$ is the set of clusters, H(w) is a single clusters entropy, N_w is the number of points in a cluster w and N is the total number of points.

Table 4.5: Cluster Evaluation

Algorithm	Distribution	Total Entropy	Cluster	Validation Label	Row Count	Entropy score per cluster
Subspace Clustering (greedy layer-wise Deep Autoencoder)	TP= 35 FP= 18	0.92445	cluster 0	TP FP	10 1	0.4394
			cluster 1	TP FP	25 3	0.4912
			cluster 2	TP FP	0 14	0

Table 4.5 evaluate the subspace clustering to cluster the candidate anomalies. The results achieved using subspace clustering are very promising and achieved better quality of clusters (lesser entropy). The score of 92% accuracy shows how well the clusters accurately corresponds to the TP and FP distribution.

Further, the information gain has been evaluated as shown in Table 4.6. Information gain is defined as the entropy of all the instances minus the weighted average of entropy for each cluster. Subspace clustering solution shows the information gain of 0.57, which depicts how well the clusters separate out true positives (TP) and false positives (FP).

Table 4.6: Information Gain

	Entropy for Candidate Anomalies	Weighted Average of cluster Entropy	Information Gain
Subspace Clustering	0.9244	0.310	0.57

4.2.2.5 Interpretability

Here, t-SNE has been leveraged to project the clusters and visualize the high dimensional vectors in space. The most relevant cluster (Anomalous rows) been identified using entropy measure has been used further to provide explainability. Set of candidate anomalies been identified from Cluster has been explained further using Kernel SHAP algorithm in Fig 4.13. This approach helps analyst in the reduction and categorization of anomalies and in finding the most relevant anomalies faster.

Table 4.7 shows the top anomalous features in detected by the trained greedy layer-wise autoencoder model. For the purpose of evaluation, anomalous rows were sorted based on the reconstruction error and anomalous cell traces were obtained for further analysis. To get a representation of the relationships between features for these top anomalous cell traces, Algorithm 3 obtained the top contributing features.

Figure 4.13: (a) Force plot to explain the predicted feature contributions. (b) Decision plot to show the set of contributing features towards the prediction with their normalized values.

Fig 4.13 review the local interpretation results generated by SHAP (normalized values). The Kernel SHAP Explainer as described in 4.2.1.5 outputs features, SHAP values and input value for anomalous rows as explained in Algorithm 3.

Table 4.7 illustrates the contribution of input features to a sample anomalous row where features with higher Effect value indicate higher impact on the entry being anomalous. This particular entry indicates that an anomalous cell trace is happening due to failed default bearer of voice caused by Mobile Management Entity (MME) Service Request Paging Attempts (**Feature 121**), MME Service Request Trigger (**Feature 127**), Initial context Setup E-RAB QCI- 5 (**Feature 143**) and MME Service Request Paging Attempts eNB (**Feature 131**).

Similar to this example, the SHAP values can help explain the impact of features to each anomalous entry detected by the proposed CCA architecture.

Table 4.7: SHapley Explanation for one Anomalous Cell Trace

Feature 121	Feature 127	Feature 143	Feature 131
Value = 4.0 Effect=**0.0097**	Value = 1.0 Effect=0.0083	Value = 5.333 Effect=0.00633	Value = 1.0 Effect=0.0059

The proposed framework prediction and the confirmation by the network engineers validated that eNB UE Context release E-RAB QCI-5 was the most important anomalous feature due to the features mentioned above.

4.2.3 Discussion on CCA Framework

In this research, an unsupervised CCA architecture has been designed to provide insights and support for the troubleshooting of network cell trace data without prior domain knowledge. This automated solution can save a lot of manual troubleshooting, with a resulting reduction in costs and resources. Results show that a deep stacked autoencoder trained using a greedy layer-wise approach to reconstruct the input data, and reconstruction error can be used to identify previously unseen anomalies with AUC of **0.9326**. This research has been carried out on Ericsson real network performance management datasets, which is their proprietary information and cannot be released in public domain. All released research results carefully avoid mention of identifiable customers or identifiable network users. While the work has been implemented and evaluated using prototype Ericsson's datasets and testing environments, the results has been demonstrated using anonymous data.

Antwarg et al. (2021) extended SHAP to explain anomalies detected by an autoencoder. The authors proposed a black-box interpretable method which aims to provide a comprehensive root cause analysis. In the framework, the top features with high reconstruction errors from trained autoencoder model were connected to the features that contribute most to reconstruction error.

Motivated by the previous work done by Chawla et al. (2020), Antwarg et al. (2021), the CCA architecture bridges the knowledge gap and offers an improved performance when an internal feedback loop was incorporated to eliminate the unimportant features (Chawla et al. 2022). Moreover, the identification of relevant clusters of anomalies based on reconstruction error can serve as a graphical interface to instance interpretation using t-SNE representation.

This novel method can be viewed as a way to explore the true/false categorization of detected anomalies without ground-truth labels as described in section 4.2.1.5. The subspace clustering when applied on the set of predicted candidate anomalies, identifies the set of clusters which correspond closely to TP and FP. The network engineer picks one instance from a cluster to determine if the cluster contains predominantly TP. Then the engineer can choose any instance from the cluster and the SHAP values for the instance are calculated.

The CCA interpretability framework based on the reconstruction error of features and SHAP values shows clear value and provide a deeper insight and causation analysis for every anomalous trace. The interpretability framework provides network operators with actionable insights which enables a deeper investigation of influencing features, thereby improving automated troubleshooting across anomalies in cell trace data using the CCA architecture. The proposed work includes a systematic, scalable evaluation of the system's utility in network performance management.

The experiments on a real-world telecom network data set demonstrate the efficacy of the proposed algorithm. The framework predicted the contributing features were indeed responsible for the anomalous cell traces. The confirmation of this by the network engineers validated the solution.

4.3 Summary

In conclusion, the answer to research question no 2 is that an reconstruction error based AD can effectively detect the anomalies and provide an interpretability to domain experts in telecommunication. This chapter introduced the algorithms which designed to identify the candidate anomalies using the feature optimised autoencoder.

The analysis of multivariant network data in order to identify anomalous data instances demonstrate the effectiveness of autoencoders (an unsupervised technique) to detect multivariant anomalies and anomalous features. To overcome the black box nature of neural networks (and thus increase their acceptance by domain experts), the application of SHapley Additive exPlanations (SHAP), are used to explain the output model of a neural network. The root cause analysis benefits from this by filtering features whose values lead (to some extent) to such anomalies.

Further, the proposed Cluster Characterized Autoencoder (CCA) architecture improves model interpretability by designing an end-to-end data driven AI-based framework. The entropy based feature ranking are clustered in reconstruction error space using subspace clustering. This clustering is seen to separate true positives and false positives and how well this is done is evaluated using entropy and information gain. A two dimensional t-SNE representation of anomaly clusters is used as a graphical interface to the analysis and explanation of individual anomalies using SHAP values. The solution provided by this unsupervised approach helps the analyst in the categorisation, identification and feature explanation of anomalies providing faster root cause analysis. Therefore, the solution provides better support for the network domain analysts with an interpretable and explainable Artificial Intelligence (AI) anomaly detection system.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This research proposes a framework that uses unsupervised DNN techniques to model and detect anomalous sequences and cell trace events in cyber security and telecommunications networks. For this reason, this thesis addressed the following research question.

"*Is it possible to design and develop high performance and interpretable anomaly detection solutions based on artificial neural networks ?* "

In designing and implementing an anomaly detection solution, certain requirements should be addressed, such as reduced computational resources and an ability to interpret model predictions. With this in mind, the following more detailed research questions have been addressed. Firstly,

Is it possible to design and develop a neural network based anomaly detection solution with reduced training and testing times, and get optimal anomaly detection results for sequential data ?

Chapter 3 evaluated the performance of an unsupervised anomaly detection model for operating system call sequences. This thesis designed and developed a neural network model to predict future sequences given that a certain existing sequence has already occurred. The primarily focus was to train a model on normal call sequences (without anomalies) using recurrent neural networks. This predicts a probability distribution for the next integer in a call sequence. Multiplying these conditional probabilities gives us an overall probability of a sequence occurring. Sequences with a low probability of occurring can be flagged as a candidate anomaly.

Stacking CNNs over RNN units handles the problem of processing sequences with thousands of steps. The replacement of LSTM with GRU layers substantially reduces the amount of training and testing times and obtains a set of results comparable to the state-of-the-art.

The reconstruction error based approach for stateful encoder and decoders demonstrated the capability to learn the sequence from both past and future directions using bidirectional autoencoders. This approach significantly improved the AUC and reached the state-of-the-art performance. This reconstruction error based approach would form the basis of an interpretability framework proposed later in the thesis.

This thesis has maintained the state-of-the-art performance for neural network models by designing computationally efficient models, with a order of magnitude smaller training times when compared to other traditional machine and deep learning models. The key findings from this research question would prove beneficial to many computer security incident response teams, which would benefit from frequent retraining to adapt to evolving typical usage. The empirical evaluation demonstrates that approach improved the overall anomaly detection performance, in terms of AUC score.

The second and main research question addressed in the thesis was *Is it possible to design a framework for Deep neural network Anomaly Detection, with enhanced interpretability and performance, to support telecommunications network troubleshooting and root cause analysis?*

Chapter 4 first addressed the above research question by designing the IUAD framework. The design of a neural network model and associated interpretation techniques provided insights and support for the troubleshooting of network cell trace data without prior domain knowledge using global and local interpretations. The level of interpretation provides network operators with actionable insights for every anomalous trace, which enables a deeper investigation of influencing features.

The Cluster Characterized Autoencoder (CCA) framework demonstrated its capability in the reduction and categorization of anomalies, which is important for network engineers in troubleshooting scenarios. Through the novel use of concepts such as entropy based feature ranking and clustering, the architecture improves performance and model interpretability by designing an end-to-end data driven AI-based framework. Results show that a deep stacked

autoencoder trained using a greedy layer-wise approach to reconstruct the input data, and reconstruction error can be used to identify previously unseen anomalies. The inclusion of an internal feedback loop improves the model performance and highlights the relevant set of anomalous features in the network.

The candidate anomalies identified using the feature optimised Autoencoder and entropy based feature ranking are clustered in reconstruction error space using subspace clustering. This clustering is seen to separate true positives and false positives and how well this is done is evaluated using entropy and information gain. This is of great benefit to telecoms engineers. By examining instances in cluster, the model achieved 92% accuracy, which shows how clusters accurately corresponds to TP and FP distribution.

A two dimensional t-SNE representation of anomaly clusters is used as a graphical interface to the analysis and explanation of individual anomalies using SHAP values. This provides support for the network domain analysts with an interpretable and explainable Artificial Intelligence (AI) anomaly detection system.

The experiments on a real-world telecom network data set demonstrate the efficacy of the proposed algorithm. The framework predicted the contributing features were indeed responsible for the anomalous cell traces. The confirmation of this by the network engineers validated the solution.

The goal of this work was to provide an interpretable framework which has been designed to provide an actionable insight in terms of developing and improving automated troubleshooting across anomalies. The unsupervised approach assist the analyst in identifying anomalies, categorizing them, and identifying their causes. This greatly speeds up the root cause analysis process. It is with firm belief, each of the above have contributed to this field, and more importantly, the results of this study will create new avenues of research.

5.2 Future Work

The novel architectures proposed in this research addresses the operating call sequences and network cell trace vulnerabilities in cyber security and telecommunication sector. However, there are unanswered questions related to sequential data interpretability. In the future, the plan is to address the problem of the location of sub-sequences causing security or performance problems in order to provide interpretability and possible root cause analysis for domain engineers. This area is a gap in the current literature and solutions are natural extensions of existing techniques and logical extensions of existing ideas and techniques. This will extend the sequential anomaly interpretation to allow human-understandable insight into the input-output data relationship.

For advanced anomaly detection in telecom, it is possible to use different application domains such as numerical (network performance indicators) and textual information (customer experience related information) to bring together data sets from different application domains. The objective is to evaluate the effectiveness of this approach to design multimodal anomaly detection based on learning across multiple network components. The future work will include applying this approach to larger datasets running on GPU machines.

Future work will focus on creating a distributed environment where data-driven solutions from various domains can be combined from different domains (RAN, EPC, IMS) into one model using mixtures of experts (Masoudnia & Ebrahimpour 2014). The data-driven solutions comprised of systematic collection, management, analysis, interpretation and application of data. The submission and acceptance to the ITU Journal titled, "AI-driven predictive and scalable management and orchestration of network slice" has been drawn from this thesis.

Another further direction of development is to develop a scalable federated learning across multiple network components and investigate Diverse Counterfactual Explanations for Anomaly Detection in Time Series (Mothilal et al. 2020). The interpretation of data-driven solutions and its application to network performance management will be an important future research area.

Bibliography

Abdi, H. & Williams, L. J. (2010), 'Principal component analysis'. doi: 10.1002/wics.101.

Abdillah, M. F., Nasri, J. & Aditsania, A. (2016), 'Using deep learning to predict customer churn in a mobile telecomunication network', *eProceedings of Engineering* **3**(2).

Abdullah Al Mamun, S. M. & Valimaki, J. (2018), Anomaly detection and classification in cellular networks using automatic labeling technique for applying supervised learning, *in* 'Procedia Computer Science', Vol. 140. doi: 10.1016/j.procs.2018.10.328.

Aggarwal, C. C. (2017), An introduction to outlier analysis, *in* 'Outlier analysis', Springer, pp. 1–34. doi: 10.1007/978-3-319-47578-3_1.

Al Mamun, S. & Beyaz, M. (2018), Lstm recurrent neural network (rnn) for anomaly detection in cellular mobile networks, *in* 'International Conference on Machine Learning for Networking', Springer, pp. 222–237.

Al Mtawa, Y., Haque, A. & Bitar, B. (2019), 'The mammoth internet: Are we ready?', *IEEE Access* **7**, 132894–132908.

Alkhayrat, M., Aljnidi, M. & Aljoumaa, K. (2020), 'A comparative dimensionality reduction study in telecom customer segmentation using deep learning and pca', *Journal of Big Data* **7**(1), 1–23.

Antwarg, L., Miller, R. M., Shapira, B. & Rokach, L. (2021), 'Explaining anomalies detected by autoencoders using Shapley Additive Explanations', *Expert Systems with Applications* **186**. doi: 10.1016/j.eswa.2021.115736.

Baldi, P. (2012), Autoencoders, unsupervised learning, and deep architectures, *in* 'Proceedings of ICML workshop on unsupervised and transfer learning', JMLR Workshop and Conference Proceedings, pp. 37–49.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. & Herrera, F. (2020), 'Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI', *Information Fusion* **58**. doi: 10.1016/j.inffus.2019.12.012.

Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. (2007), Greedy layer-wise training of deep networks, *in* 'Advances in Neural Information Processing Systems'. doi: 10.7551/mitpress/7503.003.0024.

Benvenuto, N. & Piazza, F. (1992), 'On the Complex Backpropagation Algorithm', *IEEE Transactions on Signal Processing* **40**(4), 967–969. doi: 10.1109/78.127967.

Birnie, C. & Hansteen, F. (2022), 'Bidirectional recurrent neural networks for seismic event detection', *Geophysics* **87**(3), KS97–KS111.

Bonald, T. & Proutiere, A. (2003), Wireless downlink data channels: User performance and cell dimensioning, *in* 'Proceedings of the 9th annual international conference on Mobile computing and networking', pp. 339–352.

Borisaniya, B. & Patel, D. (2015), 'Evaluation of Modified Vector Space Representation Using ADFA-LD and ADFA-WD Datasets', *Journal of Information Security* **06**(03). doi: 10.4236/jis.2015.63025.

Bosneag, A.-M., Handurukande, S. & Wang, M. (2016), Fighting fire with fire: Survey of strategies for counteracting the complexity of future networks management, *in* '19th International ICIN Conference-Innovations in Clouds, Internet and Networks', pp. 37–44.

Bradley, A. P. (1997), 'The use of the area under the ROC curve in the evaluation of machine learning algorithms', *Pattern Recognition* **30**(7). doi: 10.1016/S0031-3203(96)00142-2.

Building Autoencoders in Keras (2016), https://blog.keras.io/building-autoencoders-in-keras.html/. Accessed: 2022-07-25.

Carletti, M., Masiero, C., Beghi, A. & Susto, G. A. (2019), Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis, *in* '2019 IEEE international conference on systems, man and cybernetics (SMC)', IEEE, pp. 21–26.

Caulfield, B. (2009), 'What's the difference between a cpu and a gpu?', https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/.

Chai, T. & Draxler, R. R. (2014), 'Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature', *Geoscientific Model Development* **7**(3). doi: 10.5194/gmd-7-1247-2014.

Chawla, A., Jacob, P., Farrell, P., Aumayr, E. & Fallon, S. (2022), Towards interpretable anomaly detection: Unsupervised deep neural network approach using feedback loop, *in* 'NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium', pp. 1–9. doi: 10.1109/NOMS54207.2022.9789914.

Chawla, A., Jacob, P., Feghhi, S., Rughwani, D., Van Der Meer, S. & Fallon, S. (2020), Interpretable unsupervised anomaly detection for RAN cell trace analysis, *in* '16th International Conference on Network and Service Management, CNSM 2020, 2nd International Workshop on Analytics for Service and Application Management, AnServApp 2020 and 1st International Workshop on the Future Evolution of Internet Protocols, IPFutu'. doi: 10.23919/CNSM50824.2020.9269108.

Chawla, A., Jacob, P., Lee, B. & Fallon, S. (2019), 'Bidirectional lstm autoencoder for sequence based anomaly detection in cyber security.', *International Journal of Simulation–Systems, Science & Technology* . doi: 10.5013/IJSSST.a.20.05.07.

Chawla, A., Lee, B., Fallon, S. & Jacob, P. (2018), Host based intrusion detection system with combined cnn/rnn model, *in* 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 149–158. doi: 10.1007/978-3-030-13453-2_12.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, *in* 'EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference'. doi: 10.3115/v1/d14-1179.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. & Bengio, Y. (2015), Attention-based models for speech recognition, *in* 'Advances in Neural Information Processing Systems', Vol. 2015-Janua.

Classification: Accuracy (2022), https://developers.google.com/machine-learning/crash-course/classification/accuracy/. Accessed: 2022-03-27.

Creech, G. & Hu, J. (2013), Generation of a new IDS test dataset: Time to retire the KDD collection, *in* 'IEEE Wireless Communications and Networking Conference, WCNC'. doi: 10.1109/WCNC.2013.6555301.

Creech, G. & Hu, J. (2014), 'A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns', *IEEE Transactions on Computers* **63**(4). doi: 10.1109/TC.2013.13.

CS294A Lecture notes, Sparse autoencoder (2011), https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf. Accessed: 2022-07-25.

Dash, M. & Liu, H. (2000), Feature selection for clustering, *in* 'Pacific-Asia Conference on knowledge discovery and data mining', Springer, pp. 110–121.

Davis, J. & Goadrich, M. (2006), The relationship between precision-recall and roc curves, *in* 'Proceedings of the 23rd International Conference on Machine Learning', ICML '06, Association for Computing Machinery, New York, NY, USA, p. 233–240. doi: 10.1145/1143844.1143874.

Deeplearning4j (2022), https://deeplearning4j.org/. Accessed: 2022-04-06.

Dhanabal, L. & Shantharajah, S. P. (2015), 'A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms', *International Journal of Advanced Research in Computer and Communication Engineering* **4**(6).

Ding, N., Ma, H., Gao, H., Ma, Y. & Tan, G. (2019), 'Real-time anomaly detection based on long short-term memory and gaussian mixture model', *Computers & Electrical Engineering* **79**, 106458. doi: https://doi.org/10.1016/j.compeleceng.2019.106458.

Dong, D., Wu, H., He, W., Yu, D. & Wang, H. (2015), Multi-task learning for multiple language translation, *in* 'Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)', pp. 1723–1732.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P. & Bengio, S. (2010), 'Why does unsupervised pre-training help deep learning?', *Journal of Machine Learning Research* **11**. doi: 10.1145/1756006.1756025.

ETSI TS 133 511 V16.4.0 (2020), https://www.etsi.org/deliver/etsi_ts/133500_133599/133511/16.04.00_60/ts_133511v160400p.pdf. Accessed: 2022-04-03.

Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T. & Mizutani, K. (2017), 'State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems', *IEEE Communications Surveys and Tutorials* **19**(4). doi: 10.1109/COMST.2017.2707140.

Fernandez Maimo, L., Perales Gomez, A. L., Garcia Clemente, F. J., Gil Perez, M. & Martinez Perez, G. (2018), 'A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks', *IEEE Access* **6**. doi: 10.1109/ACCESS.2018.2803446.

Flanagan, K., Fallon, E., Connolly, P. & Awad, A. (2017), NetFlow anomaly detection though parallel cluster density analysis in continuous time-series, *in* 'Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)', Vol. 10372 LNCS. doi: 10.1007/978-3-319-61382-6_18.

Forrest, S., Hofmeyr, S. A., Somayaji, A. & Longstaff, T. A. (1996), Sense of self for unix processes, *in* 'Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy'. doi: 10.1109/secpri.1996.502675.

François Chollet (2018), 'Deep learning with python & keras'.

François Chollet (2022), 'Why use Keras - Keras Documentation', https://keras.io/. Accessed: 2022-04-07.

Fu, R., Zhang, Z. & Li, L. (2017), Using LSTM and GRU neural network methods for traffic flow prediction, *in* 'Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016'. doi: 10.1109/YAC.2016.7804912.

García-López, P., García-Marín, V., Martínez-Murillo, R. & Freire, M. (2010), 'Cajal's achievements in the field of the development of dendritic arbors', *International Journal of Developmental Biology* **54**(10). doi: 10.1387/ijdb.093001pg.

Garcia, S., Grill, M., Stiborek, J. & Zunino, A. (2014), 'An empirical comparison of botnet detection methods', *computers & security* **45**, 100–123.

Gates, C. & Matthews, P. (2014), Data is the new currency, *in* 'ACM International Conference Proceeding Series', Vol. 15-18-September-2014. doi: 10.1145/2683467.2683477.

Gers, F. A., Schmidhuber, J. & Cummins, F. (2000), 'Learning to forget: Continual prediction with LSTM', *Neural Computation* **12**(10). doi: 10.1162/089976600300015015.

Goodall, J. R., Ragan, E. D., Steed, C. A., Reed, J. W., Richardson, G. D., Huffer, K. M., Bridges, R. A. & Laska, J. A. (2018), 'Situ: Identifying and explaining suspicious behavior in networks', *IEEE transactions on visualization and computer graphics* **25**(1), 204–214.

Grosse, R. (2017), 'Lecture 15: Exploding and vanishing gradients', *University of Toronto Computer Science* .

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S. & Turini, F. (2019), 'Factual and Counterfactual Explanations for Black Box Decision Making', *IEEE Intelligent Systems* **34**(6). doi: 10.1109/MIS.2019.2957223.

Gulati, P., Sharma, A. & Gupta, M. (2016), 'Theoretical Study of Decision Tree Algorithms to Identify Pivotal Factors for Performance Improvement: A Review', *International Journal of Computer Applications* **141**(14), 19–25. doi: 10.5120/ijca2016909926.

Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*,, O'Reily Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Accessed: 2022-04-07.

Haider, W., Hu, J. & Xie, M. (2015), Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems, *in* 'Proceedings of the 2015 10th IEEE Conference on Industrial Electronics and Applications, ICIEA 2015'. doi: 10.1109/ICIEA.2015.7334166.

Hanley, J. A. et al. (1989), 'Receiver operating characteristic (roc) methodology: the state of the art', *Crit Rev Diagn Imaging* **29**(3), 307–335.

Hecht-Nielsen, R. (1992), Theory of the backpropagation neural network, *in* 'Neural networks for perception', Elsevier, pp. 65–93.

Hochreiter, S. (1998), 'The vanishing gradient problem during learning recurrent neural nets and problem solutions', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **6**(02), 107–116.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long Short Term Memory. Neural Computation', *Neural Computation* **9**(8).

Horvitz, E. (2022), 'Applications for artificial intelligence in department of defense cyber missions', https://blogs.microsoft.com/on-the-issues/2022/05/03/artificial-intelligence-department-of-defense-cyber-missions/.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* '32nd International Conference on Machine Learning, ICML 2015', Vol. 1.

Jain, A., Mao, J. & Mohiuddin, K. (1996), 'Artificial neural networks: a tutorial', *Computer* **29**(3), 31–44. doi: 10.1109/2.485891.

Kamath, U., Liu, J. & Whitaker, J. (2019), *Deep learning for NLP and speech recognition*, Vol. 84, Springer.

Keerin, P., Kurutach, W. & Boongoen, T. (2012), Cluster-based KNN missing value imputation for DNA microarray data, *in* 'Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics'. doi: 10.1109/ICSMC.2012.6377764.

Kim, J., Kim, J., Thi Thu, H. L. & Kim, H. (2016), Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection, *in* '2016 International Conference on Platform Technology and Service (PlatCon)', pp. 1–5. doi: 10.1109/PlatCon.2016.7456805.

Kingma, D. P. & Ba, J. L. (2015), Adam: A method for stochastic optimization, *in* '3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings'.

Kisačanin, B. (2017), Deep learning for autonomous vehicles, *in* '2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL)', IEEE, pp. 142–142.

Kosoresow, A. P. & Hofmeyr, S. A. (1997), 'Intrusion detection via system call traces', *IEEE Software* **14**(5). doi: 10.1109/52.605929.

L., P., E., H. & H., L. (2004), 'Subspace Clustering for High Dimensional Data: A Review', *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining* **6**(1).

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11). doi: 10.1109/5.726791.

Li, H., Ota, K. & Dong, M. (2018), 'Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing', *IEEE Network* **32**(1). doi: 10.1109/MNET.2018.1700202.

Li, Y. & Yang, T. (2018), *Word Embedding for Understanding Natural Language: A Survey*, Springer International Publishing, Cham, pp. 83–104. doi: $10.1007/978-3-319-53817-4_4$.

Lundberg, S. M. & Lee, S. I. (2017), A unified approach to interpreting model predictions, *in* 'Advances in Neural Information Processing Systems', Vol. 2017-December.

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., Liston, D. E., Low, D. K.-W., Newman, S.-F., Kim, J. et al. (2018), 'Explainable machine-learning predictions for the prevention of hypoxaemia during surgery', *Nature Biomedical Engineering* **2**(10), 749.

Lv, S., Wang, J., Yang, Y. & Liu, J. (2018), 'Intrusion Prediction with System-Call Sequence-to-Sequence Model', *IEEE Access* **6**. doi: 10.1109/ACCESS.2018.2881561.

Macha, M. & Akoglu, L. (2018), 'Explaining anomalies in groups with characterizing subspace rules', *Data Mining and Knowledge Discovery* **32**(5). doi: 10.1007/s10618-018-0585-7.

Malhotra, P., Vig, L., Shroff, G. M. & Agarwal, P. (2015), Long Short Term Memory Networks for Anomaly Detection in Time Series, *in* 'ESANN'.

Malware Capture Facility Project (2013), https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html/. Accessed: 2022-07-25.

Marteau, P. F. (2019), 'Sequence Covering for Efficient Host-Based Intrusion Detection', *IEEE Transactions on Information Forensics and Security* **14**(4). doi: 10.1109/TIFS.2018.2868614.

Martinez, E., Fallon, E., Fallon, S. & Wang, M. (2015), CADMANT: Context Anomaly Detection for MAintenance and Network Troubleshooting, *in* 'IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference'. doi: 10.1109/IWCMC.2015.7289222.

Masoudnia, S. & Ebrahimpour, R. (2014), 'Mixture of experts: a literature survey', *Artificial Intelligence Review* **42**(2), 275–293. doi: https://doi.org/10.1007/s10462-012-9338-y.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. & Kavukcuoglu, K. (2016), Asynchronous methods for deep reinforcement learning, *in* 'International conference on machine learning', PMLR, pp. 1928–1937.

Molnar, C. (2019), *Interpretable Machine Learning*, https://www.lulu.com/. https://christophm.github.io/interpretable-ml-book/.

Montavon, G., Samek, W. & Müller, K.-R. (2018), 'Methods for interpreting and understanding deep neural networks', *Digital Signal Processing* **73**, 1–15. doi: https://doi.org/10.1016/j.dsp.2017.10.011.

URL: *https://www.sciencedirect.com/science/article/pii/S1051200417302385*

Moradi, M. & Samwald, M. (2021), 'Post-hoc explanation of black-box classifiers using confident itemsets', *Expert Systems with Applications* **165**. doi: 10.1016/j.eswa.2020.113941.

Mothilal, R. K., Sharma, A. & Tan, C. (2020), Explaining machine learning classifiers through diverse counterfactual explanations, *in* 'Proceedings of the 2020 conference on fairness, accountability, and transparency', pp. 607–617.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve Restricted Boltzmann machines, *in* 'ICML 2010 - Proceedings, 27th International Conference on Machine Learning'.

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C. & Xiang, B. (2016), Abstractive text summarization using sequence-to-sequence RNNs and beyond, *in* 'CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings'. doi: 10.18653/v1/k16-1028.

Ng, A. Y. (2004), Feature selection, l 1 vs. l 2 regularization, and rotational invariance, *in* 'Proceedings of the twenty-first international conference on Machine learning', p. 78.

Park, D. C., El-Sharkawi, M. A., Marks, R. J., Atlas, L. E. & Damborg, M. J. (1991), 'Electric Load Forecasting Using An Artificial Neural Network', *IEEE Transactions on Power Systems* **6**(2). doi: 10.1109/59.76685.

Patcha, A. & Park, J.-M. (2007), 'An overview of anomaly detection techniques: Existing solutions and latest technological trends', *Computer networks* **51**(12), 3448–3470.

Ribeiro, M., Lazzaretti, A. E. & Lopes, H. S. (2018), 'A study of deep convolutional auto-encoders for anomaly detection in videos', *Pattern Recognition Letters* **105**, 13–22.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), "why should i trust you?" explaining the predictions of any classifier, *in* 'NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session'. doi: 10.18653/v1/n16-3020.

Rokach, L. & Maimon, O. (2005), 'Clustering Methods Data Mining and Knowledge Discovery Handbook', *Data Mining and Knowledge Discovery Handbook* .

Rong, X. (2009), 'R package deepnet', https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/. Accessed: 2022-08-08.

Rush, A. M., Chopra, S. & Weston, J. (2015), 'A Neural Attention Model for Abstractive Sentence Summarization', *Proceedings of EMNLP 2015* **1509**(685).

Saito, T. & Rehmsmeier, M. (2015), 'The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets', *PLoS ONE* **10**(3). doi: 10.1371/journal.pone.0118432.

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K. & Müller, K.-R. (2019), *Explainable AI: interpreting, explaining and visualizing deep learning*, Vol. 11700, Springer Nature.

Sejnowski, T. J. (2018), *The Deep Learning Revolution*, The MIT Press. https://mitpress.mit.edu/books/deep-learning-revolution(visited 2022-07-24).

Shafique, U. & Qaiser, H. (2014), 'A comparative study of data mining process models (kdd, crisp-dm and semma)', *International Journal of Innovation and Scientific Research* **12**(1), 217–222.

Shapley, L. S. (2016), *17. A Value for n-Person Games*, Princeton University Press, pp. 307–318. doi: doi:10.1515/9781400881970-018.

Shen, D., Wu, G. & Suk, H.-I. (2017), 'Deep learning in medical image analysis', *Annual review of biomedical engineering* **19**, 221.

Shen, J., Wang, R. & Shen, H. W. (2020), 'Visual exploration of latent space for traditional Chinese music', *Visual Informatics* **4**(2). doi: 10.1016/j.visinf.2020.04.003.

Shen, Y., Huang, P. S., Gao, J. & Chen, W. (2017), ReasoNet: Learning to stop reading in machine comprehension, *in* 'Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Vol. Part F1296. doi: 10.1145/3097983.3098177.

Shi, M., Wang, K. & Li, C. (2019), A C-LSTM with word embedding model for news text classification, *in* 'Proceedings - 18th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2019'. doi: 10.1109/ICIS46139.2019.8940289.

Shin, S. Y. & Kim, H.-j. (2020), 'Extended autoencoder for novelty detection with reconstruction along projection pathway', *Applied Sciences* **10**(13), 4497.

Specht, D. F. (1991), 'A General Regression Neural Network', *IEEE Transactions on Neural Networks* **2**(6). doi: 10.1109/72.97934.

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R. & Chen, J. (2020), 'DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System', *Secur. Commun. Networks* **2020**, 1–8890306.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014), Sequence to sequence learning with neural networks, *in* 'Advances in Neural Information Processing Systems', Vol. 4.

Tan, P.-N. (2009), *Receiver Operating Characteristic*, Springer US, Boston, MA, pp. 2349–2352. doi: $10.1007/978-0-387-39940-9_569$.

Trinh, H. D., Giupponi, L. & Dini, P. (2019), Urban Anomaly Detection by processing Mobile Traffic Traces with LSTM Neural Networks, *in* 'Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks workshops', Vol. 2019-June. doi: 10.1109/SAHCN.2019.8824981.

Vaidya, R., Trivedi, D., Satra, S. & Pimpale, M. (2018), Handwritten character recognition using deep-learning, *in* '2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)', IEEE, pp. 772–775.

Van der Maaten, L. & Hinton, G. (2008), 'Visualizing data using t-sne.', *Journal of machine learning research* **9**(11).

Vittaldev, V., Russell, R. P., Arora, N. & Gaylor, D. (2012), Second-Order Kalman Filters using multi-complex step derivatives, *in* 'Advances in the Astronautical Sciences', Vol. 143.

Vora, L. J. (2015), 'Evolution of mobile generation technology: 1g to 5g and review of upcoming wireless technology 5g', *International journal of modern trends in engineering and research* **2**(10), 281–290.

Wang, M. X. & Handurukande, S. (2018), 'Anomaly Detection for Mobile Network Management', *INTERNATIONAL JOURNAL OF NEXT-GENERATION COMPUTING* **9**(2).

Wang, X., Jiang, W. & Luo, Z. (2016), Combination of convolutional and recurrent neural network for sentiment analysis of short texts, *in* 'COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers'.

Werbos, P. J. (1990), 'Backpropagation Through Time: What It Does and How to Do It', *Proceedings of the IEEE* **78**(10). doi: 10.1109/5.58337.

Xie, M., Hu, J. & Slay, J. (2014), Evaluating host-based anomaly detection systems: Application of the one-class svm algorithm to adfa-ld, *in* '2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)', pp. 978–982. doi: 10.1109/FSKD.2014.6980972.

You, C., Robinson, D. P. & Vidal, R. (2016), Scalable Sparse Subspace Clustering by Orthogonal Matching Pursuit, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2016-Decem. doi: 10.1109/CVPR.2016.425.

Yousefi-Azar, M., Varadharajan, V., Hamey, L. & Tupakula, U. (2017), Autoencoder-based feature learning for cyber security applications, *in* '2017 International Joint Conference on Neural Networks (IJCNN)', pp. 3854–3861. doi: 10.1109/IJCNN.2017.7966342.

Yu, X.-H., Chen, G.-A. & Cheng, S.-X. (1995), 'Dynamic learning rate optimization of the backpropagation algorithm', *IEEE Transactions on Neural Networks* **6**(3), 669–677. doi: 10.1109/72.377972.

Zhang, A., Zhao, X. & Wang, L. (2021), Cnn and lstm based encoder-decoder for anomaly detection in multivariate time series, *in* '2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)', Vol. 5, pp. 571–575. doi: 10.1109/ITNEC52019.2021.9587207.

Zhang, M., Rajbhandari, S., Wang, W. & He, Y. (2018), DeepCPU: Serving RNN-based deep learning models 10x faster, *in* '2018 USENIX Annual Technical Conference (USENIX ATC 18)', USENIX Association, Boston, MA, pp. 951–965.

URL: *https://www.usenix.org/conference/atc18/presentation/zhang-minjia*

Zhou, C. & Paffenroth, R. C. (2017), Anomaly detection with robust deep autoencoders, *in* 'Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '17, Association for Computing Machinery, New York, NY, USA, p. 665–674. doi: 10.1145/3097983.3098052.