Network Traffic Anomaly Detection with Deep Learning

MSc Practicum Part-2
MSc Cyber Security

Christy Alex
Student ID: 23160055

School of Computing
National College of Ireland

Supervisor: Mr. Rohit Verma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Christy Alex		
Student ID:	23160055		
Programme:	MSc Cyber Security	**Year:**	2024-25
Module:	MSc Practicum Part-2		
Supervisor:	Mr. Rohit Verma		
Submission Due Date:	29-01-2025		
Project Title:	Network Anomaly detection using Autoencoder Models		
Word Count: 7711	**Page Count**:22		

Signature: Christy Alex

Date: 29-01-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Network Anomaly detection using Autoencoder Model

Christy Alex
23160055

Abstract

The increasing rate in cyber threats makes a considerable threat to traditional security features, which makes anomaly detection a crucial step in defending the network from threats. Conventional rules-based Intrusion detection system and Intrusion prevention systems most time fails to detect and prevent latest and evolving threats, which asks for the adoption of more intelligent and adaptive techniques. This research explores the implementation of a machine learning based approach based on autoencoder models to detect the malicious traffic in the network by analysis the usual working the network. The autoencoder model is trained exclusively on the regular traffic, allowing it to distinguish between regular and malicious traffic. The Research through test results suggests the use of encoder models with better thresholding for anomaly detection a autoencoder model configured to work alone would only give reduced efficiency and accuracy.

1 Introduction

The world is going though a digital transformation the inventions and improvements in technology that was introduced in the past decade was never like before inventions like quantum computing and advancements in Graphics Processing Units (GPU) and specialized chips allowed computes to perform actions though to be impossible to achieve before. World have become more and more interlinked where safeguarding infrastructure became paramount concern. Cyber-attacks like phishing and Distributed Denial-of-Service (DDoS) posing critical threat to security mechanisms imposed. The application of machine learning which advanced predominantly in the past decade is a promising way to defend such a threat. A Traditional rules-based methods of prevention always false in short to detect attacks because they are excessively depending on predefined signatures and while struggle to evolving nature of network traffic. This study determines the relative efficiency of using an autoencoder based model to determine the nature of the traffic.

The research question is:
How effectively can autoencoders trained on non-anomalous traffic identify anomalies in live network data?

The study is intended to determine the relative advantage of an encoder-based model in defending malicious traffic. Objectives of the papers also include development of a deep learning-based anomaly detection model using autoencoders, test the model in live environment to see its practical applicability. This work expected to contribute to fill the gap between the static anomaly detection and adaptability needed for the dynamic nature to threats.

The paper is structured as follows:

https://www.kaggle.com/datasets/vidhikishorwaghela/synthetic-network-traffic

Section	**Topic**
Section 2	Related works
Section 3	Research methodology
Section 4	Design Specification
Section 5	Implementation
Section 6	Evaluation
Section 7	Conclusion

2 Related Work

This section gives you a critical analysis about some of the important past works of research done in this field and advancements make in each. This analysis is focused on how each research the detection of anomalies in traffic and performance achieved by each of the researchers.

2.1 Anomaly detection with Supervised Learning methods

A supervised learning model as the name explains need the manual supervision or intervention in training the machine. The model depends on labelled datasets to classify anomalies. Datasets like NSL-KDD and CICIDs are available for training. These datasets are providing high accuracy when used with machine learning model like Convolutional Neural Network (CNN) and Support Vector Machines (SVM) when sufficient labelled data is available in both the categories. Deep learning models like Decision Trees offers more understanding and interpretability to the results. Weaknesses to this learning method are chances of overfitting and high dependency on the labelled data for the training. Accumulating such labelled data is a time consuming and depleting task while covering all attack types is not feasible since cyber security is a real of evolution and rapid changes. Using a supervised model would fail in detect unfamiliar attack types it is trained to determine. Malhotra et al. (2015)

The paper Anomaly Detection in Network Traffic using K-mean clustering is written by R. Kumari, Sheetanshu, M. K. Singh, R. Jha and N.K. Singh. This paper explores through the use of k-means clustering as a method for detecting network intrusions. The authors say that a traditional signature-based intrusion detection system would have long response time, and it won't be able to detect unknown attacks. The paper proposes a clustering-based approach using Apache spark which is a fast and scalable cluster computing framework to prove the effectiveness of the method. The methodology involves five steps which R data preparation, K means clustering, choosing the value of K, visualization, labelling and entropy. The KDD 1999 Data set is used on Apache spark to preprocess the initial stage. A K-means algorithm it's implemented in spark the group the data points into distinct groups. The optimal value for K meet represents the number of clusters determined and is chosen on the next phase. The visualization part is implemented with the use of R to gain valuable insights about the structure of the distribution. Labelling and entropy are later implemented to evaluate the quality of the clustering. The paper concludes by showing the effectiveness of the k-means clustering model in the data set which greatly detects the anomalies. The paper proposes to extend to other domains like financial data analysis and market basket analysis.

The paper "Network Traffic Anomaly Detection Using Recurrent Neural Networks" presents the application of Long Short-Term Memory networks by Benjamin J. Radford et al. for detecting anomalies that may appear in network traffic. The authors have underlined the weakness of traditional intrusion detection systems based on predefined rules and signatures and proposed unsupervised learning leveraging Long Short Term Memory (LSTM) to model the usual patterns of network communication. They tokenize and compress netflow data into sequences to enable the LSTM to learn the semantic structure of traffic. The prediction error from the model acts like an anomaly score, with higher errors indicating a likelihood of anomalies. On the ISCX IDS dataset, the LSTM model gives an impressive AUC of 0.84 for unsupervised attack identification, hence proving that it works well even when trained on mixed normal and attack traffic. The authors point out some directions for future research: exploring other deep learning architectures and the development of real-time anomaly detection systems.

2.2 Anomaly detection with unsupervised learning methods

The first paper “A Basis Evolution Framework for Network Traf c Anomaly Detection” is authored by Hui Xia, Bin Fang, Matthew Roughan, Kenjiro Cho, and Paul Tune and published in 2018 by Computer Networks. This paper research on the topic Network Traffic Anomaly Detection. The aim of the paper is to find a more accurate and adaptive method to detect anomaly that could be able to evolve over time to adapt to changes happening on the nature of the network traffic pattern. The main motivation for this research is nothing but limitations in the current anomaly detection methods. The papers say most of the current methods are based on static basic functions which limits the ability of adopt to the nature of changing traffic. The papers introduce a new framework that allows a more general function that represents the network traffic which are constrained by invariant properties like diurnal and weekly cycles.

The main contribution of the paper is Basic Evolution framework which is a more general class of function to represent the network data and an adaptational mechanism that allows the evolution of representations. The paper also suggests techniques to represent a group of anomaly points to a single anomaly point event for the sake of classification and identification.

The paper follows a methodology that includes data cleaning process to get rid of the anomalies in data, a basic generation and update process to initialize a basic function and update the function in the event of time respectively. Robust principal component analysis, orthogonal matching pursuit and principal component analysis are some of the methods the paper uses to evaluate the method generated. The experimental part of the paper includes usage of both synthetic and real-world data. The main finding in the paper is that a Basis Evolution would outperform the other methods in term of accuracy and rate of false alarms.

The paper concludes with suggesting finding a more efficient algorithms for Basis Generation and update to handle more complex anomalies. Application of the paper includes in improving the IDS and giving more effective tools to network managers to better detect and analyse the network problems. The paper also finds to connect other related research in anomaly detection like works on subspace anomalies and multi-scale methods of detection.

This paper "Network Traffic Anomaly Detection Using Recurrent Neural Networks" by Benjamin J. Radford et al. covers the application of Long Short-Term Memory (LSTM) recurrent neural networks to unsupervised anomaly detection in network traffic. The authors avoid the limitations of traditional signature-based methods by tokenizing and compressing netflow data into sequences of "words" in a way that enables the LSTM to learn the semantic structure of network communication. The prediction error of the model is taken as an anomaly score, and therefore higher errors mean higher possibility of being anomalous. As the paper demonstrates with ISCX IDS dataset, one can achieve an AUC of 0.84 for unsupervised attack identification using this approach. Further areas of research that authors will go ahead with include using other deep learning architectures, doing real-time systems, and multi-scale methodologies for anomaly clustering.

The later paper “An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection” is written by Ren-Hung Hwang, Min-Chun Peng, Chien-Wei Huang, Po-Ching Lin, and Van-Linh Nguyen and published in 2020 by IEEE Access. This paper explores an unsupervised deep learning model to detect early anomalies in the IoT networks. The motivation to conduct this research is the limitations that exist in the current detection methods related to anomalies That are predefined in nature which doesn't depends on the nature of the network. This paper introduces D-pack, which is novel based deep learning model that learns about the packet from analyzing the first few nodes of each packet. The paper highlights the importance of early detection of anomaly for better prevention and volume of processing. The detection methodology mentioned is a three-step process which includes preprocessing, traffic profile building and anomaly detection. A CNN model is used to automatically learn about the traffic in the traffic profile building phase. Later an auto encoder model uses the core for the anomaly detection phase. The datasets used in the experiment includes USTC-TFC2016, Mirai-RGU, and Mirai-CCU. The performance is calculated in terms of accuracy, precision, recall, and f1 score. The paper concludes by suggesting exploration of new techniques in detecting anomalies using a multiscale method.

The paper "Network Traffic Anomaly Detection using Machine Learning Approaches" by Kriangkrai Limthong and Thidarat Tawsook reviews the performance of some machine learning algorithms, namely naive Bayes and k-nearest neighbor, in detecting anomalies within network traffic. The authors pinpoint the shortcomings of traditional signature-based methods that are usually unable to detect new kinds of attacks and point out that statistical-based approaches can learn from experience under changing conditions. Their methodology includes data collection, feature engineering focused on interval-based features, and training the algorithms on the dataset. Tests show that K-Nearest Neighbors (KNN) generally surpasses naive Bayes. The paper emphasizes feature selection, which increases the performance of the model, and identifies further research on hybrid and deep learning models to bring improvement in detection in complex network conditions.

The paper, "Anomaly Detection in Network Traffic Using Unsupervised Machine Learning Approach," by Aditya Vikram and Mohana highlights the use of unsupervised learning model isolation forest to detect anomalies in network traffic which especially concentrated on intrusion detection. Key contributions of the research are use of unsupervised learning for anomaly detection which address the limitation of traditional rule-based detection questions to be replaced by an unsupervised learning model. The paper introduces isolation forest classification to detect the malicious traffic. Data is preprocessed and features are engineered to handle class imbalance redundancy. The paper uses evaluation metrics such as anomaly score and AUC score do access the performance of the model. The data set used in the research is KDD Cup 1999. The final stage in the methodology is visualization to gain valuable insights keyboard performance. The paper claims to have achieved an AUC score of 98.3% and is able to detect common attacks like DDoS. The research suggests several improvements like the use of a hybrid model that combines supervised and unsupervised learning techniques and practical research in feature selection and feature normalization techniques to improve the model accuracy. It also specifies the advantage of deep learning techniques in anomaly detection that can handle continuously arriving data streams. Overall, the paper provides valuable insights do the field of intrusion detection to the implementation of random forest classification to detect anomalies in the network traffic.

The paper "Network Traffic Anomaly Detection using Machine Learning Approaches" by Kriangkrai Limthong and Thidarat Tawsook explores the use of machine learning algorithms to detect anomalies in network traffic. They compare two well-known algorithms, naive Bayes and k-nearest neighbours, to identify effective interval-based features for different types of anomalies and determine the best algorithm for specific network scenarios. The authors highlight the limitations of traditional signature-based methods and the need for more robust statistical-based approaches to detect novel attacks. Their research focuses on the relationship between interval-based features and anomalies, aiming to provide insights for researchers and network administrators in selecting appropriate features and algorithms for their systems.

The paper “Network Traffic Anomaly Detection via Deep Learning," by Konstantina Fotiadou, Terpsichori-Helen Velivassaki, Artemis Voulkidis, Dimitrios Skias, Sofia Tsekeridou, and Theodore Zahariadis explores the use of a deep learning model in network traffic anomaly detection. The authors use pfSense which is an opensource firewall software to collect and analyze logs to identify threats. This paper proposes a deep learning architecture supported by CNN and Long Short-Term Memory Networks (LSTM) to detect threads and alerts in the network. This paper suggests SME supervised learning techniques with labeled and unlabeled data to improve accuracy of that anomaly detection. The paper follows a five-step methodology including data acquisition, data preprocessing, model training, anomaly detection and evaluation. The model developed shows us high accuracy (over 97%) in classifying network log. The paper represents the contribution of deep learning in the field of intrusion detection systems in network traffic.

The paper titled "Arima Model for Network Traffic Prediction and Anomaly Detection" by H. Zare Moayedi and M.A. Masnadi-Shirazi is research that explores the use of ARIMA model for network traffic predictions and anomaly detection. The authors declares that since network traffic this highly dynamic nature network provisioning and problem diagnostics most often are crucial tasks. This paper adopts a methodology on isolating anomalies from normal traffic variations to gain better predictions. The paper uses an ARIMA Model fortnight work traffic prediction which is effective to predict both malicious and normal traffic. Unlike many others this paper suggests classification of signal traffic into two parts predictable normal variation that follows it pattern and unpredictable anomalies that are pattern less. Three major predictability measures are introduced in the paper that are MSE, NMSE and SER to measure the accuracy of ARIMA model in network traffic. The authors have also done a simulation showing a random process and ARIMA process. This simulation is used to evaluate the ability of ARIMA model in detecting outliers and to identify potential troubles. The key improvement found in this paper is that the model developed is capable of detecting volume of anomalies or outliers in network traffic. Overall newspaper is a valuable contribution to the field of network traffic and anomaly detection that Concentrates on efficiency of an ARIMA model.

2.3 Anomaly Detection with Other methods

This paper by "PCA-Based Network Traffic Anomaly Detection" is written by Meimei Ding and Hui Tian, which proposes a Principal Component Analysis (PCA) -based method in detecting anomalies in network traffic . The main motivation for the project is lack of traditional methods that struggle to detect high volume network traffic. This PCA-based method that is used here can effectively analyze the higher-dimensional data by extracting the key features and reducing dimensionality. This model is peculiarized by having higher accuracy and efficiency for single-node and multi-node anomalies alike. The authors also suggest that more research is needed to combine PCA based methods with other techniques for more robust detection and analysis.

The authors of the paper "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic" are Matthew V. Mahoney and Philip K. Chan. This paper introduces a novel IDS called PHAD which relies on detection of network anomalies by analyzing the packet headers of each packet in the network unlike traditional methods like IP address and port number related. PHAD works in a way this it learns the range of values for 33 packet header field across various protocols and then assign score to each of the packet the travels based on the probability for the packet to be malicious. This strategy is tested with 1999 DARPA intrusion detection dataset and found two have detected most of the malicious packets. The authors suggest to how suggests making more research to be done on single pass version of PHAD for better tokenization and techniques.

Paper “Learning Rules for Anomaly Detection of Hostile Network Traffic," by Matthew V. Mahoney and Philip K. Chan suggests the use of LERAD which is an algorithm to detect anomalies in network traffic. LERAD is designed to identify rare events in network traffic and learn rules associated with that. The key motivation for the other two developing such a model is the inability of novel based models in detecting fast spreading worms. This model was designed to detect attacks from various sources like HTTP, SMTP and DNS. The design phase of the LERAD algorithm consists of two phases which are rule generation and rule training and validation. For future work the authors suggest the room improvements including development of a single pass version of LERAD and research into better tokenization techniques.

Paper "Network Traffic Analysis based on Collective Anomaly Detection," by Mohiuddin Ahmed and Abdun Naser Mahmood space on the need of accurate network anomaly detection systems to address challenges of denial-of-service attacks in network. The orders follows in method of collective anomaly detection with x-mean mean clustering which identifies the anomaly classic cluster of anomalies having similar nature. The efficiency of this approach is validated with DARPA dataset, which showed greater improvements in detecting denial of service attacks. The paper concludes by concentrating on the need to have more advanced clustering algorithms and multi-scale ones for handling complex anomalies

"Network Traffic Anomaly Detection," by Hong Huang, Hussein Al-Azzawi, and Hajar Brani, it's giving a complete lecture on non-signature-based anomaly detection in network traffic. It follows 3 approaches PCA-based, sketch-based, and signal-analysis-based. This paper also talks about a framework to integrate these 3 approaches. In the paper many times the authors stress the need to increase the importance of network security and improved mobility detection systems. This paper reading served as a valuable insight to understand deeply about various anomaly detection systems and its implementations.

Federico Simmross-Wattenberg, Juan Ignacio Asensio-Pérez, Pablo Casaseca-de-la-Higuera, Marcos Martín-Fernández, Ioannis A. Dimitriadis, and Carlos Alberola-López and published “Anomaly Detection in Network Traffic Based on Statistical Inference alpha-Stable Modeling” in 2011 at IEEE transactions on Dependable and Secure Computing. Study aims to find is robust and accurate but there to detect the dirt traffic anomalies. The main motivation for this research is the limitations in existing anomaly detection methods that are most often depends on gamma distribution and Poisson model. This paper introduces a new anomaly detection Mora based on alpha-stable distributions and statistical hypothesis testing. The practical involves 4 steps data acquisition, data analysis, inference and Validation. In the inference stage A generalized likelihood ratio test (GLRT) is used to classify the traffic into anomaly or normal. The validation phase is characterized using Receiver Operating Chatacteristic (ROC) curves. This method Is shown to be outperforming the traditional anomaly detection methods in terms of accuracy and detection rates. The paper suggests the use of more robust and efficient algorithms for alpha-stable parameter estimation. Potential applications after research includes improved accuracy and reliability of intrusion detection systems and empower the network managers with more advanced tools for detection.

3 Research Methodology

The research approach adopted for the study on which this thesis is based is outlined below by describing the processes, techniques, and tools utilized. This section also entails the analysis and assessment of scientific soundness of the data collected.

3.1 Research Process

Objective: To detect anomalies in network traffic using autoencoders.
Steps followed:

1. **Literature Review**: The study of related work has been extended to identify proper methodologies for setting a baseline.
2. **Data Collection**: The collection of network traffic data for normal and malicious traffic was obtained using publicly available datasets like 'synthetic_network_traffic' and live network capture tools like Suricata in a virtual environment.
3. **Data Preprocessing**:
   - Filtered normal traffic for autoencoder training.
   - Encode categorical features into numerical representations for protocol type, service, and flags.
   - Scaled numerical features (e.g., bytes, packets) using Standard Scaler to ensure uniformity.
4. **Model Design**: Designed the PyTorch deep autoencoder model that is supposed to regenerate normal traffic.
5. **Training**: The Autoencoder is trained only on normal traffic data to minimize the reconstruction error.
6. **Testing** the model by utilizing mixed data against anomaly thresholds defined by reconstruction errors.
7. **Validated**: The model's performance is compared to baseline methods.

3.2 Apparatus Used

Software:

- Programming Language: Python.
- Libraries: PyTorch, NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn.
- Dataset Used: Synthetic Network Traffic Anomaly Detection Dataset, Live collected Dataset from Suricata.

Hardware:

- Host Machine running the Windows 11.
  Specifications:
  - Processor: Intel i5 CPU.
  - RAM: 16 GB.
  - Storage: SSD, 500 GB.
- Virtual Machine running Kali Linux
  Specifications:

Storage:50GB
RAM Allocated: 8 GB
No of Processors Allocated: 8 CPU

3.3 Data Collection and Preparation

Source:

- Publicly available datasets containing labelled traffic data labelled Is Anomaly or regular
- Captured live network traffic with tools like Suricata.

Data Size: Approximately 1 million records of network traffic flow from public dataset and 50,000 sample live collected form Suricata tool.

Preprocessing Techniques:
Cleaning:

- removal of missing irrelevant data.
- Encoding of categorical data as one hot vectors.
- Normalization of numerical features for better model performance.
- Normalization of Anomalies to numerical features.

3.4 Experimental Arrangement

Training Scenario:
Dataset Split: 80% Training, 10% Validation and 10% Test Sets. .
Training of the model is exclusively done on normal traffic as encoders learn from normal traffic to detect anomaly traffic.
Threshold Selection:
Threshold on the reconstruction error set while analysing the distribution in the validation data.
Live Deployment Test:
Developed model is tested on live virtual environment collecting live data from the network to classify the nature of the network.

3.5 Statistical Methods

Reconstruction Error Analysis:

- Computed Mean Squared Error for each record to understand the variation from the original and computed.
- Cutoff threshold statistically predetermined such as the 95th percentile.

Performance Indicators:

- Precision, Recall, F1 score on anomaly detection.
- Receiver Operating Characteristic (ROC) Curve and Area Under Curve (AUC) to measure trade-offs between true and false positives.

3.6 Data Analysis

Outlier Detection:

- Outliers detected when reconstruction error outperformed the threshold.
- Checked the feature-feature correlation to identify anomalies.

Visualization:

- Reconstruction error distribution plotted.
- ROC curve for understanding model performance.
- Precision-recall chart for highlighting anomaly detection efficiency.

3.7 Summary of Procedure

- Data Collection and Preprocessing.
- Model Design and Training with Normal Traffic data.
- Anomaly detection based on the reconstruction error.
- The statistical techniques of performance evaluation. It ensures the scientific undertaking is strong, evaluative, and comprehensively validates the proposed solution through a methodological approach.

4 Design Specification

4.1 Techniques and Architecture

This implementation involves the use of autoencoder neural networks, which are designed for anomaly detection in network traffic data. The below image shows the architecture of the model designed in the paper. Architecture is having and incoming component and an outgoing component for incoming and outgoing traffic respectively. The architecture follows layered approach where the encoder model is placed at the innermost location directly interacting with the corporate network. The encoder would direct the genuine traffic to the inside network while blocking the malicious ones. The outgoing traffic works in a similar way just before the traffic is sent to the outside world the traffic is passed through the model to detect malicious nature of traffic.

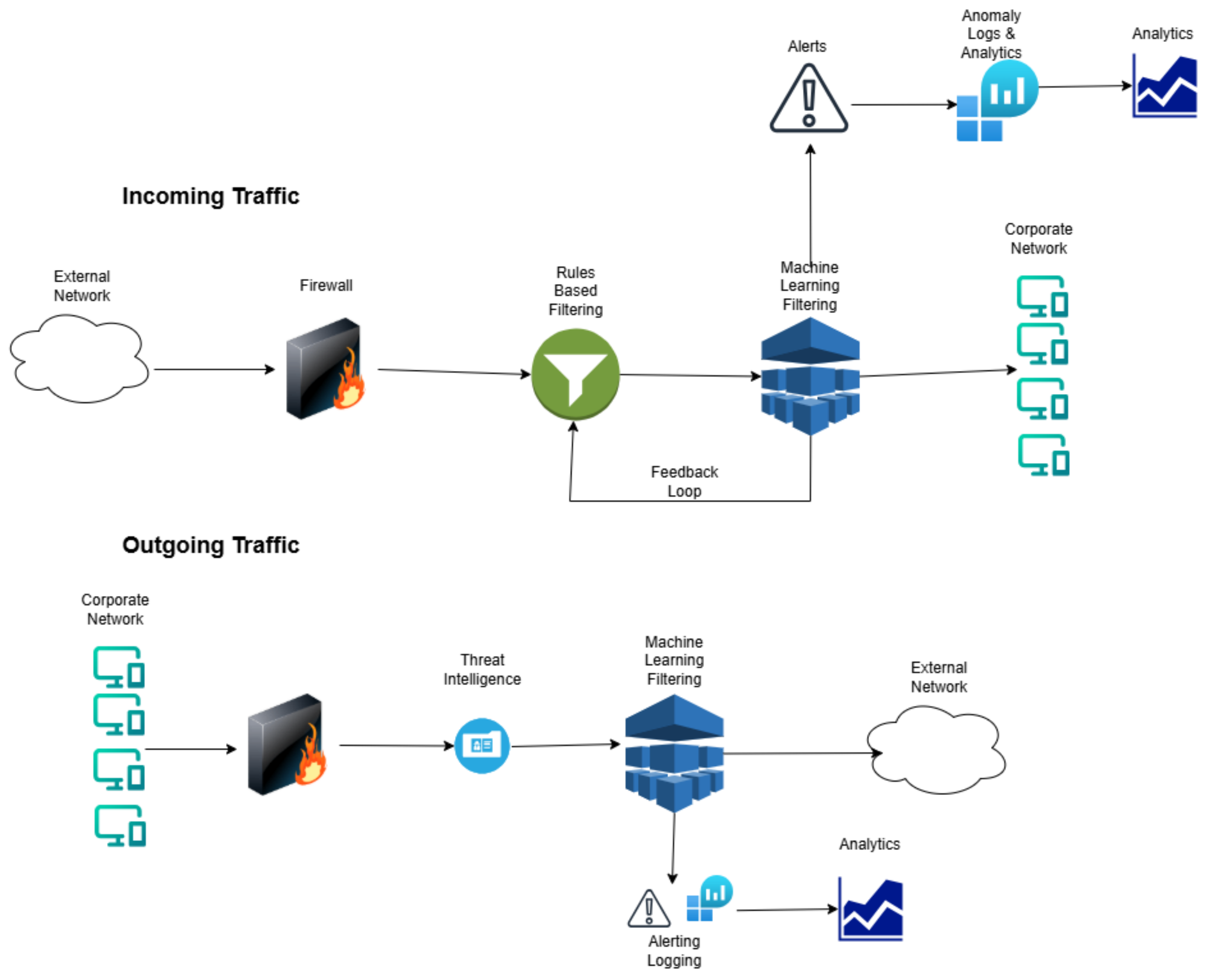


Fig 1: Architecture of the Model

1. Autoencoder Neural Network Architecture:

- **Encoder**: This model compresses the input data into a lower-dimensional latent representation via several fully connected layers mixed with LeakyReLU and Batch Normalization, speeding up the training and further stabilizing it.
- **Latent Space**: Comprises the most representative features of the input data so that the model will be able to reconstruct normal patterns effectively.
- **Decoder**: This reconstructs the input data from the latent representation and structurally mirrors the encoder.
- **Output**: The source input is compared with a source-the recreated output-to compute the reconstruction error subsequently used for anomaly detection. See the image below to see the general structure of the encoder model.

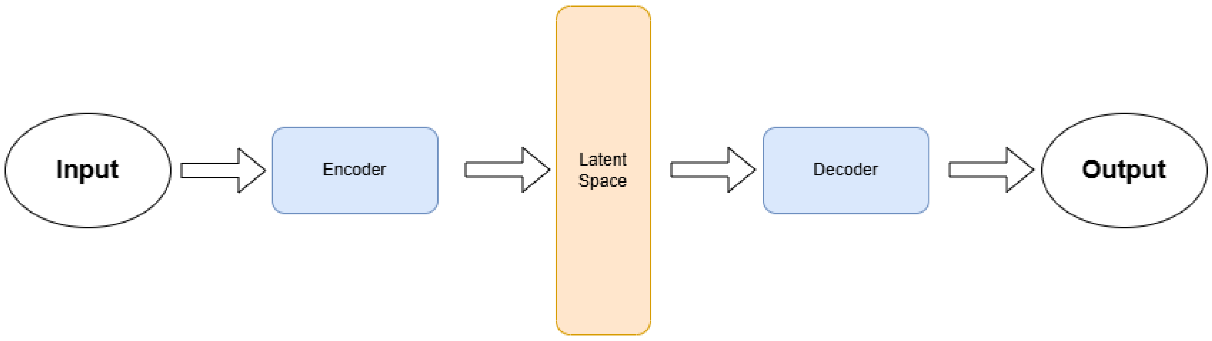


Fig 2: Structure of an encoder model.

An encoder model has 5 stages beginning from Input ending at Output. An encoder and decoder would come in between the latent space which does majority of the classification work.

2. Regularization Techniques:

- Dropout: This is applied both in the encoder and decoder to prevent overfitting.
- Weight Decay: Used in optimization to perform L2 regularization.

3. Structure:

It is implemented using PyTorch: a deep learning framework that allows flexible design and training of desired neural network architectures.

4. Learning Strategy:

- Adam Optimizer: It is used for smooth convergence with efficient gradient descent at lower learning rates.
- Learning Rate Scheduler: Dynamically adjusts the learning rate to prevent overshooting or stagnation during training.
- Early Stopping: This stops the training in case the model performance doesn't improve for some number of epochs.

4.2 Requirements

4.2.1 Data Requirements:

Input data consists of network traffic records with numeric and categorical features. Categorical features are encoded using Label Encoding, whereas numeric features are brought to a standard scale using Standard Scaler.

4.2.2 Hardware Requirements:

A system with a GPU is recommended for speeding up training because of the model's deep architecture.
Current GPU: GTX 1650

4.2.3 Software Requirements:

- Python 3.x.
- Libraries: PyTorch, NumPy, pandas, scikit-learn, and joblib.

4.2.4 Performance Indicators:

- Reconstruction Error: The performance of the model is measured as MSE between the input and the output.
- Anomaly Detection Threshold: It is defined as a mean reconstruction error plus three standard deviations of the error distribution. Algorithm Description

4.3 The autoencoder works as illustrated:

Input Processing:
This section pre-processes the network traffic data: it cleans up anomalies, encodes categorical variables, and scales numerical features.
Encoding:
The input data are fed into the encoder to reduce the dimensionality, which captures the salient features in the compressed latent space.
Decoding: The decoder projects this latent representation back into the original input dimensions in such a way that it attempts to reconstruct the data like the original one.
Loss Calculation: The reconstruction error, or MSE, may be determined by comparing the reconstructed data against the original input.
Anomaly Detection: Reconstruction errors are analysed for anomalies after training. Errors that exceed a predefined threshold signify anomalous behaviour. This design will ensure robustness, scalability, and effectiveness in anomaly detection while considering all the developments related to deep learning and regularization techniques of such complex network traffic data.
Threshold Calculation: A Threshold values is calculated to decide if the reconstruction error is higher than the threshold to decide if the traffic is malicious or not.

*Threshold = mean + 3 * standard deviation*

mean – Average of the reconstruction errors form the training.
Standard deviation - Standard deviation of the reconstruction errors form the training.
Threshold – Reconstruction error value above which considered an anomaly.

The image below shows the classification process based on the threshold of reconstruction error using an encoder model. When the threshold of reconstruction error is more than an allowed level the model is blocked as anomaly while others are considered regular.

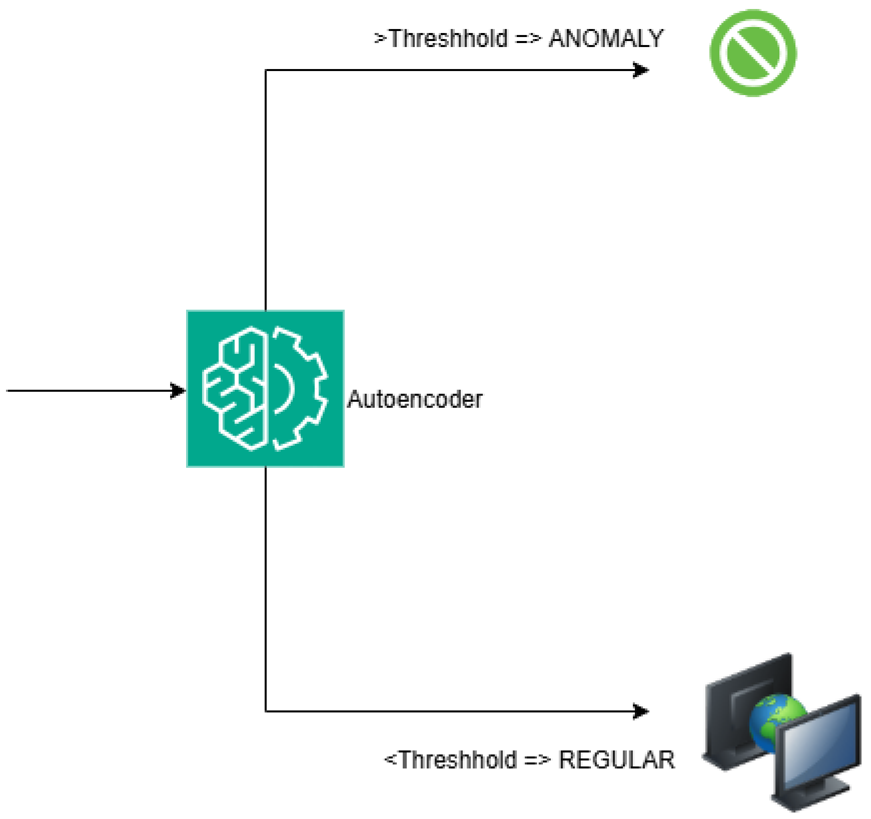


Fig 3: Classification Based on threshold.

5 Implementation

5.1 Outputs Produced

Trained Autoencoder Model:

- Built and trained in PyTorch.
- Outputs the reconstruction errors for every input instance to classify the network traffic as normal or anomalous.

Transformed Dataset:

- Pre-processed network traffic, including scaled numeric features and one-hot encoded categorical ones.
- Analysed for anomalies by applying reconstruction error thresholds.

Visualisations:

- Graphs of reconstruction error distribution in normal and anomalous traffic.
- ROC-Receiver Operating Characteristic curve for the performance evaluation.
- Precision-recall curve highlighting detection efficiency.
  Fig : ROC Curve on Model

5.2 Tools and Languages Used

Programming Language: Python.
Libraries and Frameworks:
PyTorch: To create and train the autoencoder.
Numpy and Pandas: for data frame manipulation and pre-processing.
Scikit-learn: for feature scaling and evaluation metrics.
Matplotlib and Seaborn: For plotting graphs and visualizations.

Environment:
Kali running on Virtual Machine for network traffic collection.
Jupyter Notebook for coding and iterative debugging.
Windows host machine for development and training purposes.

5.3 Process of Implementation

Data Preprocessing
Cleaning and transforming the raw network traffic data into a model input format.
The features are standardized using StandardScaler; the categorical columns are one-hot encoded.
Autoencoder Model Development:
Implemented the deep autoencoder using PyTorch, with fully connected layers.
The encoder was configured to compress data into a latent space, while the decoder reconstructed data into its original format.
Model Training:
It was then trained only on the normal traffic data using an MSE loss function.
It has subsequently been optimized using Adam's optimizer with a learning rate of 0.001.
Anomaly Detection:
This was later used in the reconstruction of test data post-training.
The reconstruction error was calculated for every single data point. After that, anomalies were defined by a threshold.
The model developed is saved to deploy the model in a live machine. A virtual machine is used to deploy the model to detect the machines traffic as anomaly or not. Bases on the result the traffic details are printed.
Evaluation and Validation:
Model performance was analysed based on precision, recall, F1 score, and the ROC Curve. Created visualizations to interpret the model performance and underline anomalies.

5.4 Final Output

Trained autoencoder weights are saved into a .pth format for future use. Encoder training is terminated if the MSE is not improving at 3 consecutive epochs. Best of the training model is used to deploy on the virtual machine from where the dataset was originally collected from. Network of the virtual machine is analysed by the model to categorize the traffic malicious or regular.

6 Evaluation

The purpose of this section is to provide a comprehensive analysis of the results and main findings of the study as well as the implications of these finding both from academic and practitioner perspective are presented. Only the most relevant results that support your research question and objectives shall be presented. Provide an in-depth and rigorous analysis of the results. Statistical tools should be used to critically evaluate and assess the experimental research outputs and levels of significance.
Use visual aids such as graphs, charts, plots and so on to show the results.

The Performance of the model is evaluated using the ROC curve and Precision-Recall curve.

6.1 ROC Curve:

It plots the sensitivity against the false positive rate. The area under the curve is the measurement of the performance of the model. The histogram for ROC performance is highly skewed to the right, the sharp peak when coming closer to zero. Here X-axis shows the Reconstruction error. The Lower values here represents that the data-point is well defined by the model here, mean while a high value suggests that it's an anomaly. Y-axis here is the number of data points given reconstruction error. The distributed shape shows that the data points have a low reconstruction error. There are also a small amount of data points having higher errors which show there were some anomalies detected. The red dashed line here marks the anomaly detection threshold which is approximately 0.0154. The threshold decided here is a statistical figure calculated by the means of standard deviations. The data points having reconstruction errors above the threshold are classified as anomalies

Implications: most data points showing low reconstruction error show that the encoder is doing a good job representing the normal traffic. But the long tail here is an indication of significant false positives. Creation existing between the bulk of the data and the threshold is not very large which means the model is not very efficient at distinguishing between the normal and anomaly data. The success rate of the anomaly detection is heavily dependent on the threshold is the threshold is too low there can be lot of false positive classifications. If it's too high, they can be many false negative volumes. The large number of data points near the threshold suggest that the model needs potentially more adjustment to the threshold to get better performance.

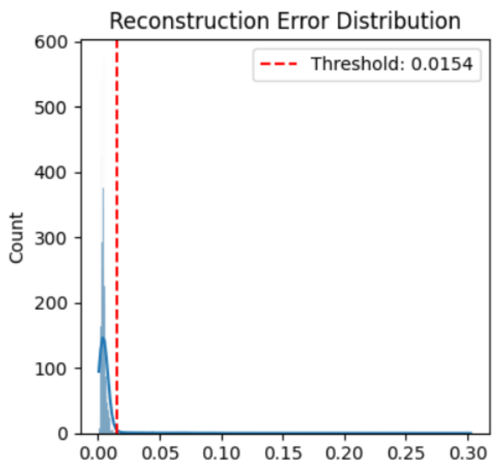


Fig 4 :ROC Performance

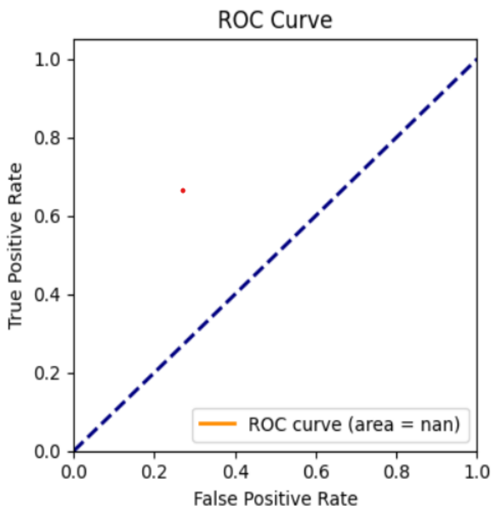


Fig 5: ROC Characteristic Cure.

The Roc curve indicates for performance. Here the X axis is the file supposed to decorate, Y axis represents the true positive rate. The single red point on the curve represents specific operating point of low true positive rate and low false positive rate.

6.2 Precision Curve

The X axis here represents the recall, which is the proportion of actual anomaly identified correctly by the model. The Y axis is the precision which is the ratio of correctly identified anomalies to all the instances of the labeled as anomalies. An ideal model would have a precision of one and the recall of one which would have a curve that stays at the top right corner. The car in the image is almost a straight line which indicates that the model's performance is very close to random guessing. The small red dots represents a specific point on the curve which is a particular threshold of low precision and moderate recall.

Implications: the near diagonal structure show that anomaly detection was not successful. There can be several reasons for having a low precision for the model which can be included for model training, issues with the data set, fault threshold selection. The encoder model may not have been properly trained on the data set so with insufficient training or learning rate or even a faulty architecture could also be the reason. The data set chosen might itself be the problem the insufficient representation of data good how hindered the model's ability to learn. Even though the threshold is visually represented in the Roc curve which also can be the reason for low precision. The image below shows an inverse relation between the precison and recall while the recall was low initially the precision was very high ,when the recall reached maximum the precision declined to a minimum rate.

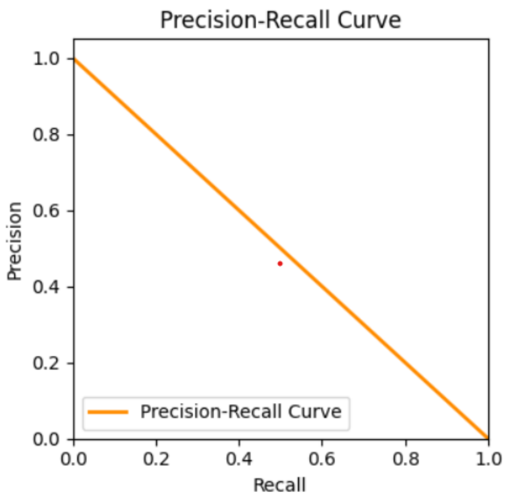


Fig 6: Precicion Curve

The encoder model has tried to be implemented with different sorts of data collected from various sources.

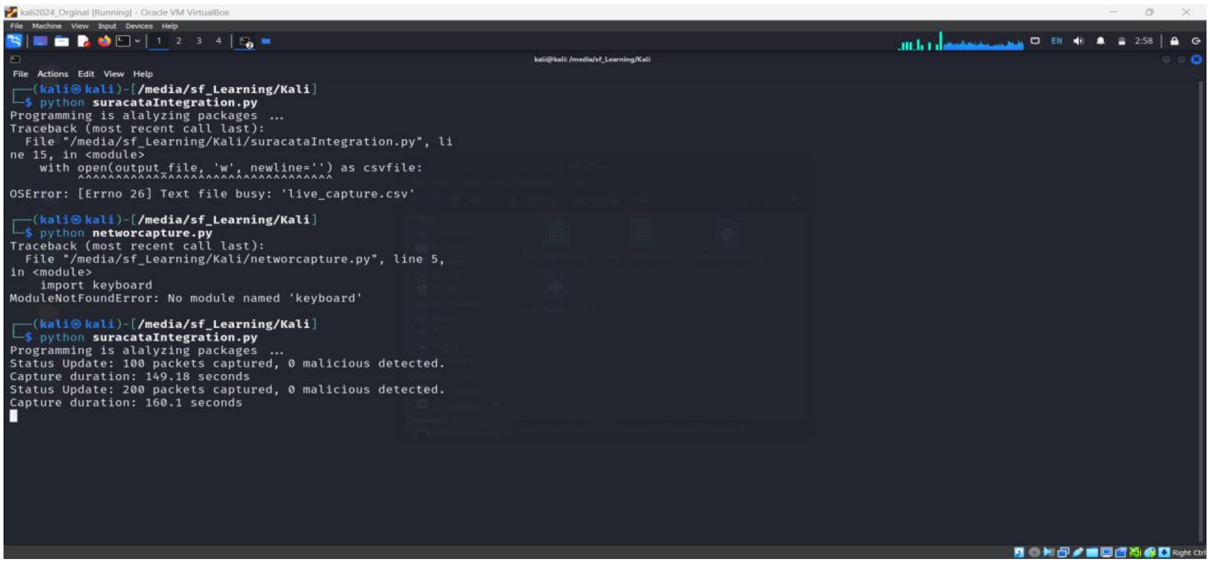
6.1 Experiment / Case Study 1

Publicly available dataset “Synthetic Network Traffic Anomaly Detection Dataset” train and test the auto encoder mortal to detect its efficiency. Features of the dataset were
Features: SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol, BytesSent, BytesReceived, PacketsSent, PacketsReceived, Duration, IsAnomaly.
The model was trained on this data set having 1,000,000 records including both malicious and regular traffic. When this model developed was deployed on the virtual machine the model utterly failed in classifying the traffic. Which was one of the main reasons experiment 2 was conducted connecting live data from tools like Suricata.

6.2 Experiment / Case Study 2

A new data set is collected from the virtual machine environment hosted by Kali using tools like Suricata. Which is an open source IDS/IPS system. Network traffic from the virtual

machine is collected using this open source tool to CSV format to train the model. Features of the data set collected includes
Features: SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol, BytesSent, BytesReceived, PacketsSent, PacketsReceived, Duration, Malicious.
Among these 11 features only 7 were used to train the encoder model. Which were “SourcePort, DestinationPort, Protocol, BytesSent, BytesReceived, PacketsSent, PacketsReceived, Duration”. The model was trained on this data set made individually showed better accuracy in categorizing live captured data in the virtual machines since the dataset it's originally collected from similar virtual machine. So it would be great to collect the data set from the same network the machine learning model is planning to be implemented.
The image below is a screenshot of how the network traffic is collected using suricate tool in kali Linux environment. The program runs in terminal and collects the date from the traffic until it is terminated by user. For this research purpose the program is allowed to run for hours colleting packets from incoming traffic.


kali2024_Orginal [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: /media/sf_Learning/Kali
File Actions Edit View Help

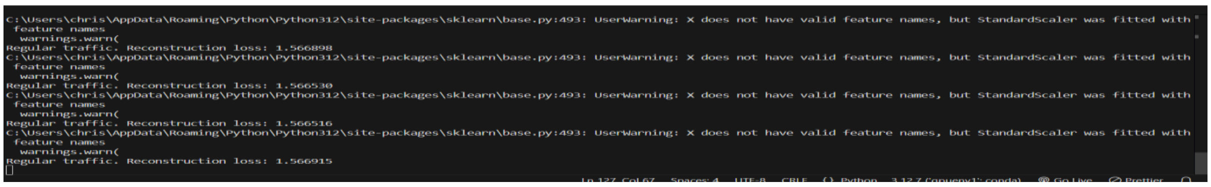
```
┌──(kali㉿kali)-[/media/sf_Learning/Kali]
└─$ python suracataIntegration.py
Programming is alalyzing packages ...
Traceback (most recent call last):
  File "/media/sf_Learning/Kali/suracataIntegration.py", li
ne 15, in <module>
    with open(output_file, 'w', newline='') as csvfile:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

OSError: [Errno 26] Text file busy: 'live_capture.csv'

┌──(kali㉿kali)-[/media/sf_Learning/Kali]
└─$ python networcapture.py
Traceback (most recent call last):
  File "/media/sf_Learning/Kali/networcapture.py", line 5,
in <module>
    import keyboard
ModuleNotFoundError: No module named 'keyboard'

┌──(kali㉿kali)-[/media/sf_Learning/Kali]
└─$ python suracataIntegration.py
Programming is alalyzing packages ...
Status Update: 100 packets captured, 0 malicious detected.
Capture duration: 149.18 seconds
Status Update: 200 packets captured, 0 malicious detected.
Capture duration: 160.1 seconds
█
```

Fig 7: Kali Virtual Machine Collecting Packages for the dataset.



```
C:\Users\chris\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with
 feature names
  warnings.warn(
Regular traffic. Reconstruction loss: 1.566898
C:\Users\chris\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with
 feature names
  warnings.warn(
Regular traffic. Reconstruction loss: 1.566530
C:\Users\chris\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with
 feature names
  warnings.warn(
Regular traffic. Reconstruction loss: 1.566516
C:\Users\chris\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with
 feature names
  warnings.warn(
Regular traffic. Reconstruction loss: 1.566915
```

Fig 8: Live network Classification in Virtual Machine.

The screenshot above shows the encoder model running in an virtual environment classifying the incoming packets. When ever a packet is captured with high reconstruction error which is greater than the threshold it would be displayed as malicious in the terminal.

6.5 Discussion

Detailed discussions of the findings of the experiments and case studies follow. The discussion critically analyses experiments, shows the limitations, and suggests ways to improve.

Critique of Experiment / Case Study 1

For the first experiment, the "Synthetic Network Traffic Anomaly Detection Dataset" is a rather diverse traffic dataset. Given that was good to test initial successes, the results being on a virtual machine were suboptimal. Key issues include:

- **Overgeneralization of Dataset:** The synthetic dataset most probably did not generalize well to the live traffic environment.
- **Deployment Environment Mismatch**: The model developed with synthetic data was poorly performing in classifying real-world traffic because the training data didn't include nuance specific to the domain.

Suggested Changes:

- Use more representative datasets of the intended deployment environment.
- Performed feature engineering to understand how real-world traffic in scenarios would feed into the model.

Critique of Experiment / Case Study 2

Results for the second experiment utilized live data obtained straight from the virtual machine environment and thus had better classification accuracy. This underlines the importance of environment-specific data. Among the limitations:

Small Training Dataset: The dataset was environment-specific and rather small as compared to the synthetic dataset.

Threshold Sensitivity: Detection performance is highly sensitive due to the choice of threshold; hence, mainly manual tuning is performed.

Suggested edits:

- This increases the size of the training dataset through the collection of longer lengths of traffic data.
- Automate threshold optimization using techniques like grid search or adaptive thresholding.

General Observations and Context with Literature

These results are consistent with prior work that has established that anomaly detection models require domain-specific training data. Various studies have pointed out that, although convenient, synthetic datasets often lack nuances of live traffic, which reduces their value in the wild. Apart from these, the literature also points to threshold setting sensitivity as a general problem with anomaly detection models.

Proposed Changes:

- Include more features or metadata from the live traffic to train on.
- Try other model architectures, including variational autoencoders, that might allow for better representation learning. Parallel studies using other datasets or other

methodologies compare findings. Addressing these limitations and incorporating the improvements suggested will allow remarkable improvements in the model's performance for practical deployment.

7 Conclusion and Future Work

This was for the study and development of a machine-learning-supported anomaly detection system of network traffic. Thus, the key question in this study was: "To what extent could an autoencoder model detect anomalies in network traffic based on reconstruction errors?" The main goals were to design, implement, and evaluate the performance of this model using synthetic and network live data, identify its strengths and limitations, or its practical applicability.

Summary of Work Done: An autoencoder model was designed and trained to answer the research question on the detection of network anomalies. Firstly, an autoencoder model was tried on a synthetic dataset, which needed to be used at the very beginning because of its high controllable nature. Secondly, cognizant of the limitation of synthetic data in nature, another experiment was done by capturing live traffic data in a virtual machine environment.

Careful feature selection was done for those relevant to anomaly detection, and performance was analysed by metrics including the ROC and precision-recall curves.

Key Findings

- **Synthetic Dataset Results**: The model seemed to perform well on the synthetic dataset it was trained and tested on, but, when actually deployed, it performed exceedingly poorly. This is one of the challenges of trying to generalize from the results of synthetic data into actual traffic.

- **Results on Live Dataset**: The model is trained with live traffic data from the same environment in which it is deployed, giving appreciable accuracy. Again, this points to the fact that environment-specific datasets could have an important say in anomaly detection.

- **Threshold Sensitivity**: This model relied heavily on the choice of threshold value while trying to detect anomalies. Inappropriate thresholds either lead to too many false positives or do not find the anomalies at all; therefore, better threshold optimization methods have to be found.

Limitations and Implications

The present study characterizes an autoencoder as a prospective tool in network anomaly detection, provided that its training is done with domain-specific data. Limitations faced by the tool include:

- **Dataset Qualit**y: The synthetic dataset lacked the complexities of real-world traffic, limiting its usefulness in practical applications.

- **Dependency of Threshold**: Besides, detection became less adaptive for differing conditions of traffic, since the threshold used was picked manually.

- **Generalization of Models**: Even on live data, the model's performance was poor for edge cases; this hints at further refinement in the architecture or training process.

These findings go along well with the existing literature on the trade-off between synthetic and real data sets, added to the challenges of fine-tuning unsupervised anomaly detection models.

Proposals for Future Work

- **Dynamic Thresholding**: Future studies may be performed to use adaptive thresholding schemes which dynamically change with traffic patterns or other statistical properties, reducing the need for manual tuning.

- **Diversified datasets**: more collection and usage of real traffic data in different environments and temporal conditions improve the generalization capability of the model.

- **Advanced Architectures**: Other architectures, such as variational auto-encoders, or combinations of both supervised and unsupervised techniques in one algorithm, can do anomaly detection more precisely and reliably.
- **Feature Enhancement:** This can be further improved by incorporating other features, such as time-series characteristics or metadata, for better detection performance.
- **Commercial Application**s: From a practical perspective, embedding the model into currently available intrusion detection systems like Suricata may lead to real-world deployment. Subsequently, this can be followed by scalability and user-friendliness during piloting in a controlled enterprise environment.

References

Chan, P. and Mahoney, M. (2001). *Scholarship Repository @ Florida Tech Scholarship Repository @ Florida Tech PHAD: Packet Header Anomaly Detection for Identifying Hostile PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic Network Traffic PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic*.

Ding, M. and Tian, H. (2016). PCA-Based Network Traffic Anomaly Detection. *TSINGHUA SCIENCE AND TECHNOLOGY*, 21(5).

Fotiadou, K., Velivassaki, T.-H., Voulkidis, A., Skias, D., Tsekeridou, S. and Zahariadis, T. (2021). Network Traffic Anomaly Detection via Deep Learning. *Information*, 12(5), p.215. doi:https://doi.org/10.3390/info12050215.

Huang, H., Al-Azzawi, H. and Brani, H. (n.d.). *Network Traffic Anomaly Detection*.

Hwang, R.-H., Peng, M.-C., Huang, C.-W., Lin, P.-C. and Nguyen, V.-L. (2020). An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access*, 8, pp.30387–30399. doi:https://doi.org/10.1109/access.2020.2973023.

Iglesias, F. and Zseby, T. (2014). Analysis of network traffic features for anomaly detection. *Machine Learning*, 101(1-3), pp.59–84. doi:https://doi.org/10.1007/s10994-014-5473-9.

Kumari, R., Singh, S., Jha, R. and Singh, N. (2016). *Anomaly Detection in Network Traffic using K- mean clustering*.

Limthong, K. (2013). Real-Time Computer Network Anomaly Detection Using Machine Learning Techniques. *Journal of Advances in Computer Networks*, pp.1–5. doi:https://doi.org/10.7763/jacn.2013.v1.1.

Mahoney, M. and Chan, P. (n.d.). *Learning Rules for Anomaly Detection of Hostile Network Traffic*.

Radford, B., Apolonio, L., Trias, A. and Simpson, J. (n.d.). *Network Traffic Anomaly Detection Using Recurrent Neural Networks*.

Simmross-Wattenberg, F., Asensio-Perez, J.I., Casaseca-de-la-Higuera, P., Martin-Fernandez, M., Dimitriadis, I.A. and Alberola-Lopez, C. (2011). Anomaly Detection in Network Traffic Based on Statistical Inference and \alpha-Stable Modeling. *IEEE Transactions on Dependable and Secure Computing*, 8(4), pp.494–509. doi:https://doi.org/10.1109/tdsc.2011.14.

Singh, R., Srivastava, N. and Kumar, A. (2021). Machine Learning Techniques for Anomaly Detection in Network Traffic. *2021 Sixth International Conference on Image Information Processing (ICIIP)*, pp.261–266. doi:https://doi.org/10.1109/iciip53038.2021.9702647.

Vikram, A. and Mohana (2020). *Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach* . [online] Available at: https://ieeexplore.ieee.org/document/9137987 [Accessed 6 Dec. 2024].

Xia, H., Fang, B., Roughan, M., Cho, K. and Tune, P. (2018). A BasisEvolution framework for network traffic anomaly detection. *Computer Networks*, 135, pp.15–31. doi:https://doi.org/10.1016/j.comnet.2018.01.025.

Zare Moayedi, H. and Masnadi-Shirazi, M. (n.d.). *Arima Model for Network Traffic Prediction and Anomaly Detection*.