ANOMALY DETECTION IN NETWORK TRAFFIC

USING DEEP LEARNING TECHNIQUES

By

Devapriya Devapriya

Submitted to

The University of Roehampton

In partial fulfillment of the requirements

MASTER OF SCIENCE IN DATA SCIENCE

ABSTRACT

The security and integrity of network systems are paramount in an increasingly interconnected digital landscape. In this study, we use the prowess of Long Short-Term Memory (LSTM)-based autoencoders to tackle the crucial problem of anomaly identification in network data. Our strategy is based on an investigation of the UNSW-NB15 dataset, a sizable collection of network traffic data that includes a wide range of legitimate and criminal activity. We create a deep learning model that makes use of LSTM layers to understand the complex temporal patterns present in network communications by taking advantage of the sequential character of network traffic data. The autoencoder architecture is used because it can recreate typical network traffic while highlighting variations that point to abnormalities. We confirm the model's effectiveness in separating anomalous behaviors from the usual through a comprehensive training and evaluation approach.Our study's examination of LSTM-based autoencoders for network anomaly detection, optimisation of model parameters, and evaluation of performance measures are its key strengths. We also cover ethical issues, making sure that private user information is protected and sensitive network data is handled responsibly. The LSTM-based autoencoder algorithm, which achieves an overall accuracy of 93.60% on the test dataset, beats other cutting-edge machine learning techniques on the UNSW-NB15 dataset. The study makes a contribution to the larger subject of cybersecurity by highlighting the possibilities for future developments in network traffic monitoring and by providing insights into the use of deep learning techniques for real-time anomaly identification.

DECLARATION

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Devapriya Devapriya Signed

Date : 06/09/2023

ACKNOWLEDGEMENTS

I'd like to take this chance to show my sincere thanks to the people and institutions who have offered constant support and direction during my MSc project and academic experience in general.

I am really grateful for the advice, knowledge, and mentorship of my first supervisor **Dr.Ogechukwu Okonor** during the course of this project. Your advice and commitment to helping me improve academically have been priceless.

I want to thank **Dr.Charles Clarke,** for your excellent instruction, which has increased my knowledge and comprehension of this project. Your enthusiasm for the topic has been genuinely motivating.

I sincerely thank **Dr.Lisa Haskel** for their interesting lectures and thought-provoking discussions on this work. It has been a pleasure working with you to create an engaging learning environment.

I appreciate all of the academic staff's efforts to create a positive learning atmosphere. Their enthusiasm for both study and instruction has served as a continual source of inspiration.

I'd like to express my gratitude to my students and peers for our friendship, teamwork, and exchange of information. My learning experience has been improved by our exchanges, which have made this academic journey special.

My family and friends' continuous support, compassion, and inspiration are much appreciated. My academic success has been motivated by their confidence in my talents.

I would like to express my appreciation to all the people and institutions that provided information, viewpoints, or input for this study. Your involvement in this initiative was essential to its success.

I would like to thank the universities, libraries, and internet databases that have made it possible for me to access a plethora of knowledge and research materials for my academic endeavors.

Thank you for your constant support and contributions to my academic path, to everyone named above and the numerous more who have contributed. My development as a learner and researcher has been irrevocably shaped by your impact.

TABLE OF CONTENTS

1. Introduction

In today's digital world and with rapid internet usage and continuous cloud technology adoption,we depend more and more on networks for a variety of daily activities. The word "network" is used to describe a computer network, which is a group of linked gadgets (such computers, servers, routers, switches, etc.) that may interact and share resources. Modern information technology is based on networks, which are widely utilized in a variety of contexts, including homes, offices, data centers, and the internet. As businesses and individuals use these networks for communication, information sharing, and economic transactions, maintaining their security becomes of utmost importance. However it has become complex to manage and monitor such a large-scale network infrastructure connectivity. Now relating this network complexity to the impending attacks or network fault users can face using a network which can be disturbing, it will then be wish to define the expectable computer network performance and its communication metric in order to easily detect the occurrence of network anomaly.
In order to protect these networks from possible threats and assaults, anomaly detection in network traffic analysis is essential.

1.1 What is an anomaly?

Network anomaly refers to any unusual or unexpected behavior inside a computer network that differs from expected patterns. Different types of network anomalies might appear, including anomalous traffic patterns, sudden shifts in network behavior, strange communication flows, unusual resource usage, and departures from standard network protocols. For the purpose of maintaining network performance, guaranteeing security, and spotting possible risks or problems, these anomalies must be found and examined. Fig1 shows a general graphical representation of anomaly with actual, expected, and deviated flow. The blue line shows the normal flow and the red lines are the abnormal flow or anomalies.Using a bound, these normal and aberrant flows are separated. The commonly recognised anomalies in network traffic analysis are behavioral anomalies, protocol anomalies, structural anomalies and network anomalies.

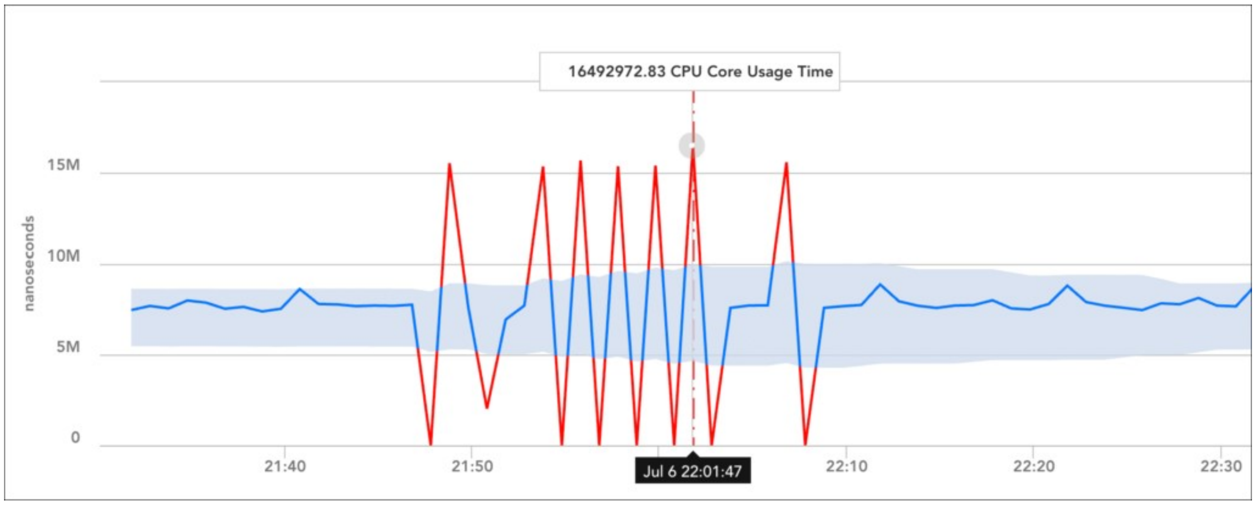


Fig1. A simple graph representation of anomaly

A behavioral anomaly is an abnormality in the behavior of a host, a user, or an application on a network. This includes behaviors that are unexpected or unusual for a certain entity, including a rapid shift in user behavior, an excessive number of network connections from a single host, or odd communication patterns between hosts.

A communication protocol that differs from the norm. In contrast to the conventional antivirus/anti-malware method, an intrusion detection system (IDS) looks for protocol irregularities to detect assaults without being aware of the malware's real pattern (signature). Anomalies in procedures can lower false positives when they are well understood.

A structural anomaly is a change in the network topology or architecture that is unanticipated or inconsistent with the configuration that has been set up. This could involve the emergence of new hardware, adjustments to network pathways, or modifications to routing schemes.

Security-related anomalies have a direct connection to security threats and assaults. Examples include brute-force login attempts, denial-of-service (DoS) attacks, port scanning, and unauthorized access to critical resources.

Network security is of the utmost significance in the linked world of today. Protecting computer networks from possible threats and assaults becomes a vital responsibility as organizations increasingly rely on them for their operations. The discovery and detection of security-related abnormalities in network traffic is one of the major issues in guaranteeing network security. Port scanning, DoS attacks, brute-force login attempts, intrusion attempts, data exfiltration, malware propagation are some kind of security based anomalies.

Network security anomalies that are connected to security present serious threats and may have negative effects on organizations. When unauthorized people gain access to the network infrastructure or crucial services, it can be caused by security-related irregularities that lead to network breaches. This may lead to malicious software being installed, network configurations being altered, or unauthorized access to sensitive data.Data that is sensitive or confidential may be lost or stolen as a result of anomalies that include data exfiltration or unauthorized transfers. Financial losses, harm to a company's reputation, problems with the law and compliance, and a decline in consumer confidence can all arise from this. Service interruptions can be brought on by DoS attacks and other abnormalities that overtax network resources. These interruptions may affect the operation and availability of crucial systems and services, resulting in prolonged downtime, lost productivity, and disgruntled clients.

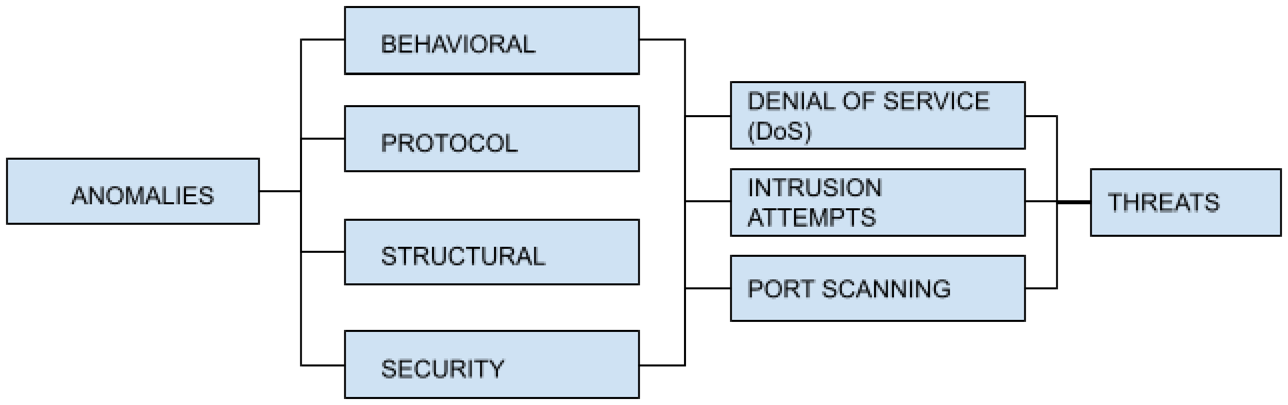


Fig2. Types of anomalies

To reduce these risks and their effects, prompt notice and action are essential when dealing with security-related abnormalities.The challenge is to identify irregularities in network traffic data in order to reduce possible cyber threats, protect the organization's vital assets, and maintain network security for companies and institutions.

Create a system that is effective and efficient for spotting unusual patterns in network traffic data. The objective is to make a contribution to the field of cybersecurity by developing a solution that may aid in increasing network security measures, minimizing the effects of security breaches, and quickly recognising possible cyber threats or assaults. Their capacity to use data science, network traffic analysis, and anomaly detection to solve real-world cybersecurity problems should be shown.

Gather pertinent network traffic data from the right sources, then prepare it for analysis by cleaning, filtering, and/or modifying the data as needed. Conduct exploratory analysis on the data to learn more about the traits, distributions, and patterns of the network traffic data. Find any anomalies or irregularities in the dataset. Feature engineering is the process of extracting educational elements from network traffic data that can identify significant trends and traits. Engineering features like flow statistics, time-based features, packet header features, or higher-level features may be involved. Based on the features of the data, evaluate various anomaly detection methodologies, such as statistical techniques, machine learning models, or deep learning techniques, and choose the most appropriate models. Create the chosen models, then train them using the preprocessed data. Use relevant evaluation measures, such as accuracy to rate the effectiveness of the built anomaly detection models. To evaluate the models' efficacy and efficiency, compare and analyze the outcomes.

1.2 LEGAL, ETHICAL, SOCIAL AND PROFESSIONAL CONSIDERATIONS

This study investigates the crucial area of anomaly detection in network traffic, which covers a wide range of legal, moral, social, and professional issues in addition to technological difficulties. It is crucial to make sure that gathering and studying network traffic data is legal. Understanding the legal system is crucial in a time when data privacy rules, like the GDPR of the European Union, have become more prominent. The rights and privacy of the people whose data may be used are respected, and this project completely complies with applicable data protection laws. In our work, ethics are of utmost importance. By its very nature, network traffic monitoring might raise moral questions about the possible violation of user privacy. We debate the moral implications of our work while recognising the necessity of openness, informed consent, and strict protections to secure the interests of all those involved.mWe must be aware of the wider societal ramifications of our activities, which go beyond the technical details of anomaly identification. Unrecognized network abnormalities have the potential to cause serious security flaws, data breaches, and cyberattacks with far-reaching societal repercussions. By correcting these anomalies, we help make the internet a safer place for consumers, companies, and society at large. Professionalism is the cornerstone of our strategy, therefore take it into consideration. Adherence to industry standards and best practices is essential for network security and anomaly detection. Our methodology, assessments, and suggestions are all based on a commitment to preserve the highest standards of professional honesty.

2. LITERATURE AND TECHNOLOGY REVIEW

2.1 LITERATURE REVIEW

Anomaly detection in network traffic has become a crucial topic of research due to the quick development of networked systems and the complexity-increasing nature of cyber threats. This review of the literature gives a general overview of recent research and developments in the area of identifying network traffic anomalies, with an emphasis on the use of machine learning and deep learning methods. The articles that were chosen demonstrate a variety of tactics and strategies for tackling this problem.

[1] Convolutional Long Short-Term Memory (C-LSTM) neural networks are used in this study by Kim and Cho to demonstrate an unsupervised deep learning method for the early identification of network traffic anomalies. The study emphasizes the significance of applying deep learning algorithms to effectively identify abnormalities in network traffic data. [2] A dynamic network anomaly detection system based on deep learning methods is proposed by Hwang et al. Their research highlights the promise of deep learning models and emphasizes the necessity for real-time detection and adaptive learning in resolving network abnormalities. [3] In this study, machine learning methods are used to detect anomalies in wireless sensor networks. The importance of customized strategies for various network types is highlighted as the authors investigate how machine learning may be applied to the particular context of wireless sensor networks.[4] A machine learning-based solution for anomaly and cyber threat identification in streaming network traffic data is presented by Poornima and Pramasivan. Their study focuses on how machine learning may be used practically to improve network security. In the paper [5] A machine learning-based method for anomaly and cyber attack identification in streaming network traffic data is presented by Komisarek et al. in this research. Their study emphasizes the value of real-time detection in reducing online dangers. [6] Machine learning methods are used to implement anomaly detection approaches, according to Rao et al. Their research offers helpful advice on how to use machine learning for network traffic analysis. [7] By examining various kinds of flow characteristics, this research offers a hybrid neural network strategy for enhancing network anomaly detection. The advantages of integrating several strategies for improved accuracy are emphasized by the writers.[8]by Hareesh et al. describes a system for detecting anomalies based on the examination of packet header and payload histograms. Their

research examines cutting-edge methods for spotting irregularities in network traffic [9]An unsupervised machine learning method for anomaly identification in network traffic is presented by Vikram and Mohana. Their work demonstrates the potency of unsupervised methods for locating network abnormalities.

2.2 TECHNOLOGY REVIEW

An essential part of network security is anomaly detection, which looks for unusual patterns or behaviors that can point to threats or attacks. Traditional approaches to anomaly detection frequently depend on statistical analysis methods, however new developments in deep learning have shown promise in terms of increasing the efficiency and accuracy of detection. This technology evaluation focuses on the deployment of a hybrid method for detecting anomalies in network data that combines statistical analysis with deep learning techniques.

In the hybrid method, statistical analysis is principally used for (a) feature engineering,which involves extracting, transforming and selecting features from raw data to depict the behavior and patterns in the network traffic that helps in finding suitable features and characteristics that can be fed to the deep learning model as input. (b) data preprocessing, guarantees that the deep learning model can accept the data. (c) establishing baselines and thresholds for typical network traffic behavior to compare the incoming data and spot the variations or abnormalities. The mean, median, standard deviation , percentile and more complex approaches like time series decomposition and forecasting are used for this purpose.

Due to the capacity to automatically uncover detailed patterns and representations from complex data, deep learning techniques have drawn interest in the field of anomaly detection. Deep learning methods, like neural networks, can capture dependencies across numerous layers and non-linear interactions. They are more scalable than certain conventional statistical techniques because they can be parallelized and optimized to handle vast amounts of data effectively. Deep learning algorithms may extrapolate from training data to create predictions about unobserved data. This means that in the context of network traffic, the model may be applied to fresh instances of network traffic and successfully detect abnormalities after it has been trained on a representative dataset. Ic can identify subtle and intricate patterns in network data that may be difficult to identify using conventional statistical techniques.

Deep learning methods are frequently represented by (a) Convolutional Neural Networks (CNNs) that are beneficial for anomaly identification in network traffic when local patterns and correlations are crucial since they are excellent at capturing spatial dependencies in data. They can spot geographic anomalies in network flows or packet contents. (b) Recurrent neural networks (RNNs), such as LSTMs or GRUs, are good at detecting temporal connections in sequential data. They perform effectively for detecting anomalies in data from sequential or time-series network traffic. (c) Autoencoders which are designed to recover input data. Departures from the output that is rebuilt after learning a compressed version of the input might signify abnormalities.

To capitalize on the advantages of both strategies, a hybrid strategy combines statistical analysis with deep learning techniques. Through this integration, network traffic anomalies can be detected more thoroughly and precisely. By extracting pertinent statistical characteristics, locating outliers, or creating baselines, statistical analysis can be used to preprocess the data. A deep learning model can then be used to further analyze these preprocessed features. Deep learning and several statistical models can be used in an ensemble manner to enhance detection performance. Deep learning models that capture complex patterns and connections can be used after statistical models to give first screening or filtering. Deep learning models can be used to improve anomaly detection in a separate domain once characteristics or patterns identified via statistical analysis in one domain are transferred to the other. This method can get around complexity or data availability issues.

The hybrid approach offers the possibility for more precise and reliable anomaly identification in network data by combining statistical analysis methods with deep learning models. Deep learning algorithms excel in capturing complicated, non-linear correlations and patterns, whereas statistical analysis offers interpretability, stability, and baseline estimate. Combining these techniques makes it easier to spot small abnormalities and minimizes false positives.

Effective model selection, training, and data preparation are essential to the hybrid technique's success. For training and assessment, it also needs a sizable volume of labeled or labeled anomaly data. To maximize the performance of the hybrid approach, much thought should be given to feature selection, model design, hyperparameter tweaking, and assessment measures.

For the purpose of detecting anomalies in network traffic, the hybrid approach that fuses statistical analysis with deep learning approaches shows significant potential. This hybrid technology, which combines the best aspects of both methods, provides more precise, reliable, and effective anomaly detection, enhancing the overall security of network infrastructures. To investigate other hybrid configurations and improve the integration for diverse network environments and security needs, more research and development is required.

3. METHODOLOGY

This project's goal is to create a network traffic anomaly detection system utilizing a hybrid approach that combines statistical analysis and deep learning. The hybrid technique is anticipated to increase accuracy and efficacy in identifying possible threats and aberrant behaviors, which is essential for network security.

A popular network security dataset for research on intrusion detection is the UNSW NB15 dataset. It was developed by the University of New South Wales (UNSW), Australia's Network Security Lab. The dataset includes a variety of regular and attack traffic types and was created to offer a realistic depiction of network traffic.In addition to regular network traffic, the dataset also contains a variety of assaults, including denial-of-service (DoS), port scanning, and intrusion attempts. Statistical attributes, flow-based features, and certain payload-specific traits are all included in the dataset's collection of features, which are retrieved from network traffic and are included in each record. For the access of the performance models, it is splitted into training and testing datasets.

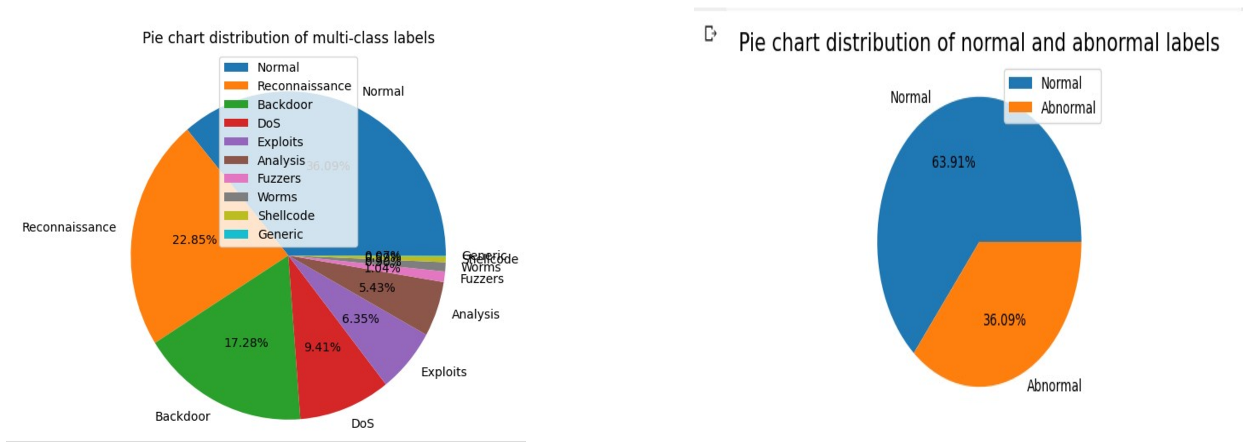


Fig3. Representation of normal and abnormal values in dataset

The UNSW-NB15 dataset must be preprocessed in order to be ready for analysis. In order to prepare the data for future analysis, it must first be cleaned, essential characteristics must be extracted, and the data must be transformed. To prevent the analysis from being skewed, the duplicate entries were eliminated. Missing values were examined and corrected by either eliminating the associated entries or imputing the proper values. The raw network traffic data has been converted from the UNSW-NB15 dataset through data preprocessing into a structured and cleaned dataset suitable for training machine learning models and carrying out anomaly detection research. The final dataset is prepared for further investigation utilizing hybrid algorithms that accurately identify anomalies in network traffic by combining statistical analysis and deep learning.

The mean and standard deviation are computed in order to determine the data's central tendency and dispersion.

$$\text{Mean}(\mu) = \frac{\Sigma(x_i)}{N} \qquad \text{..... (1)}$$

$$\text{Standard deviation}(\sigma) = \sqrt{\frac{\Sigma(x_i - \mu)^2}{N}} \qquad \text{.....(2)}$$

where x_i is the individual data points and N is the total number of data points

Then, the standardized scores, or Z-scores (Z), are calculated to express how much a data point deviates from the mean in terms of standard deviations, assisting in the detection of abnormalities.

$$\text{Z-Score}(Z) = \frac{(x_i - \mu)}{\sigma} \qquad \text{.....(3)}$$

where x_i, μ, σ are individual data points, mean and standard deviation respectively.

In order to examine how network events are timed, inter-arrival times are calculated. By measuring the number of packets, one may measure the volume of traffic and identify sudden variations in activity. Unusual usage patterns for protocols and ports are discovered through distribution analysis. The distribution of packet sizes is then evaluated to look for any potential network assaults or anomalies. These statistical traits give us important information about how networks behave. Unusual behavior in a feature or set of characteristics can be found using univariate and multivariate analysis. By using time series analysis, anomalies based on temporal trends and patterns can be found. Clustering is used to discover outlier flows that do not fit into any clusters and to set baselines for typical behavior. Using departures from typical patterns, outlier identification can further draw attention to probable abnormalities.By combining these approaches, it allows to detect anomalies in network data and capture different facets of network traffic behavior. This strategy is frequently employed in practical anomaly detection systems, where many approaches complement one another and increase detection rates while lowering the number of false positives. However, each approach must be carefully chosen and tailored depending on the features of your particular dataset and the anomalies you hope to identify.

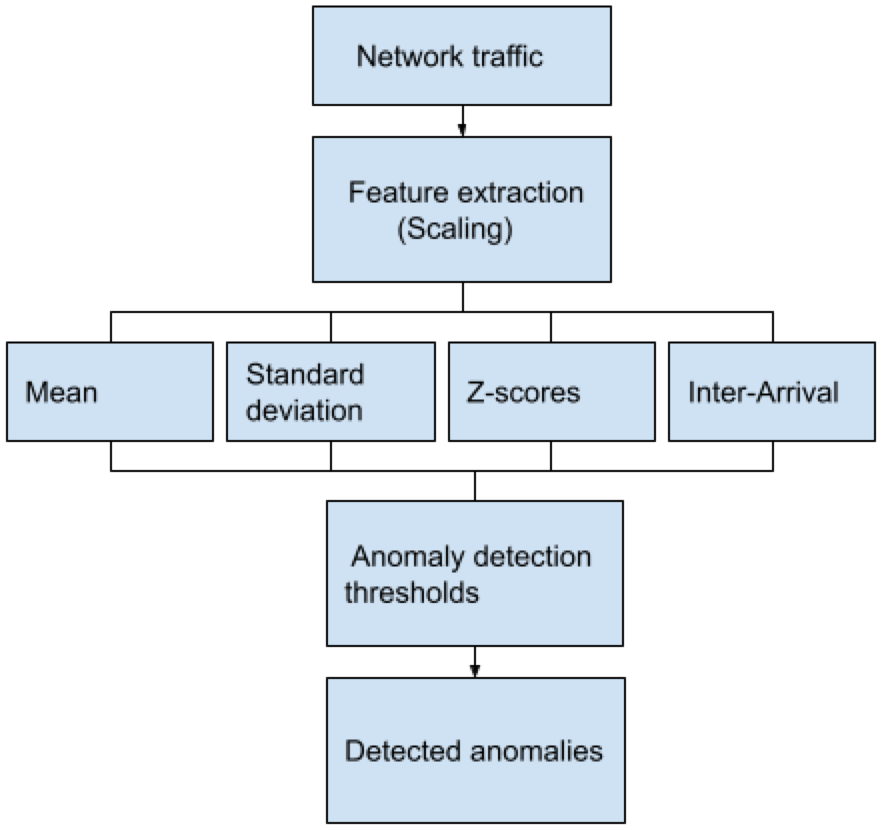


Fig4 : Statistical Analysis of Network Traffic Data

Making a good anomaly detection system requires careful consideration when choosing a deep learning model. The selection procedure entails comparing many deep learning architectures and selecting the one that is most appropriate for the data's properties and the particular anomaly detection job. Because it can effectively capture temporal relationships in sequential network traffic data, an LSTM-based autoencoder was used in this instance.Let's explore why an LSTM-based autoencoder is suitable for this project: (a) Since network traffic data is made up of flows or packets that arrive over time, it is by its very nature sequential. Since LSTM (Long Short-Term Memory) networks are created primarily for sequential data, they are excellent at identifying temporal relationships and patterns. (b) autoencoders are designed to learn a compressed representation of the input data (encoding) and then reconstruct (decode) the original data from the compressed form. The autoencoder works well for anomaly detection since it makes an effort to reduce the reconstruction error. An LSTM-based autoencoder can be trained to efficiently capture the typical patterns in the data and learn to reflect the sequential nature of network traffic data in its encoding layer. (c) Since the number of packets in a flow might change, LSTM networks can manage variable-length sequences, which is advantageous for network traffic data. The LSTM-based autoencoder can capture long-term relationships in sequences of different lengths thanks to its adaptability. (d) Memory cells in LSTMs are capable of storing and retrieving data from earlier time steps. The LSTM-based autoencoder can recognise anomalies based on how they differ from prior patterns thanks to its memory feature. The model is a good choice for spotting abnormalities in network traffic because of its capacity to capture temporal relationships, unsupervised nature, and efficacy in identifying departures from the usual patterns. However, effective hyperparameter tuning, architectural design, and the availability of a relevant dataset for training all contribute to the model's performance.

Long short-term memory (LSTM), often known as RNN architecture, is one sort of recurrent neural network. LSTMs are ideally suited for managing sequences of data over time since they are intended to handle the problem of disappearing or exploding gradients that might arise in standard RNNs. The key benefit of LSTM is that they can keep long-term dependencies and store information for extended periods of time, which enables them to retain information from the start of a sequence and continue it through to the finish. This is crucial for tasks requiring sequential data, including time series analysis, speech recognition, natural language processing, and more. The four gates that make up an LSTM memory cell are input gate, output gate, forest gate and self-recurrent neuron. What additional information should be added to the cell state is decided by the input gate. It generates a value between 0 and 1 for each component of the cell state after receiving input from St-1 and Xt. The LSTM cell's ultimate output is determined by the output gate. It receives input from St-1 and Xt and outputs a value in the range of 0 and 1, scaling the cell state to create the final hidden state ht. This gate chooses which data to remove from the cell state. It generates a value between 0 and 1 for each component of the cell state given the current input (Xt) and the previous hidden state's input (St-1).

$f_t = \sigma(XtUf + St\text{-}1Wf + bf)$ ....(4)

i_t=σ(XtUi+St−1Wi+bi) ....(5)

o_t=σ(XtUo+St−1Wo+bo) ....(6)

C̃t=tanh(XtUc+St−1Wc+bc) ....(7)

Ct = Ft * Ct-1 + It * C̃t ....(8)

ht = Ot * tanh(Ct) ....(9)

Where, (Uf, Ui, Uo,Uc), (Wf, Wi, Wo, Wc), (bf, bi, bo, bc) represents input weights, recurrent weights and biases respectively. At time step t, Xt, St, and Ct represent the input, hidden, and cell states, respectively. The hidden and cell states at time step t-1 are represented by St-1 and Ct-1, and σ represents sigmoid activation.

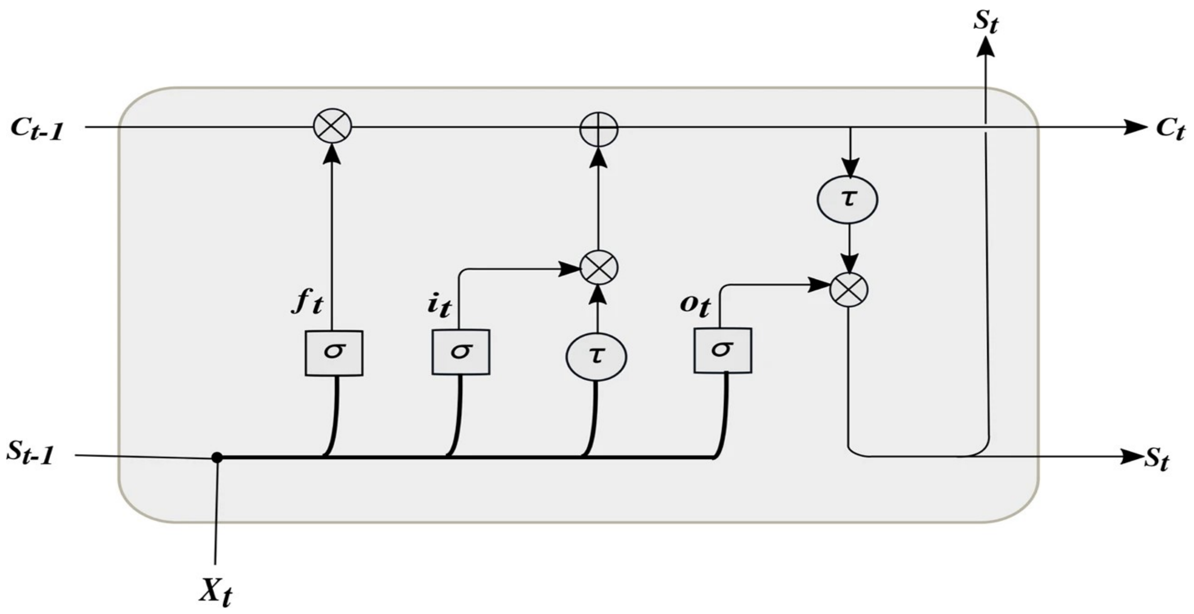


Fig5. The LSTM block where ft, it, ot are forget, input, and output gates respectively.

An artificial neural network called an autoencoder uses data reconstruction to learn effective representations of input data. It falls under the umbrella of dimensionality reduction techniques and is a member of the family of unsupervised learning algorithms. An autoencoder's main goal is to encode the input data into a lower-dimensional representation (latent space), which it will later decode back into.The autoencoder architecture consists of encoder and decoder. A compressed representation of the input data is created in the latent space by the encoder. Multiple layers, usually fully connected layers or other neural network layers, are used in this mapping process to gradually lower the dimensionality of the input data. The encoder's last layer displays the input data's compressed form or latent code. The encoder's compressed representation (also known as the latent code) is used by the decoder to recreate the original data. The decoder, like the encoder, is made up of several layers that gradually increase the dimensionality back to the initial data space. The rebuilt data, which strives to be as near to the original data as possible, is what the decoder's final output reflects.

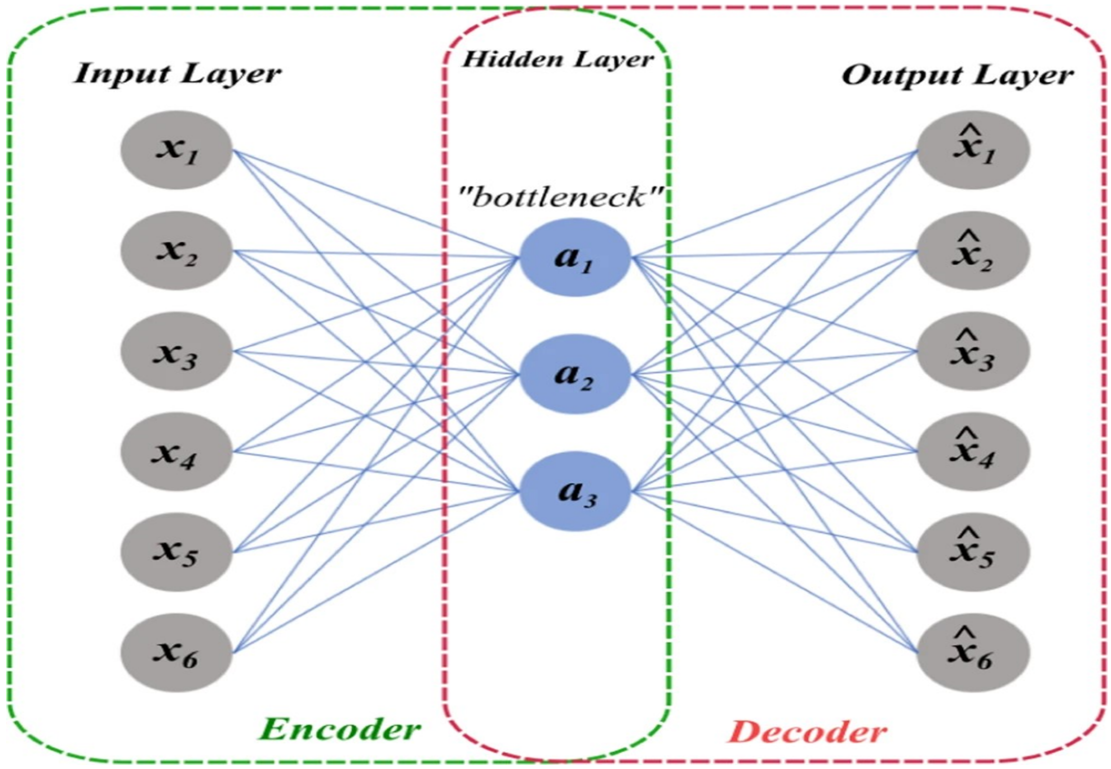


Fig 6. Autoencoder architecture

The two stages may be expressed as follows given:

An unlabeled input dataset x_n , where n = 1, 2,..., N and x_nRm

h(x)=f(W1x+b1) ....(10)

$\hat{x}$=g(W2h(x)+b2) ....(11)

Where $\widehat{x}$ is the decoder (or reconstruction) vector of the output layer and h(x) is the hidden encoder vector derived from input vector x. In addition, f is the encoding function, g is the decoding function, W1 and W2 are the encoder and decoder's respective weight matrices, and b1 and b2 are the phase-specific bias vectors.

Reconstruction error is the term used to describe the discrepancy between the original input and the reconstructed input (or output). This reconstruction error is modeled as an objective function, and the model attempts to minimize it during training, for example, by minimizing $||x-\widehat{x}||2$. Multiple AE layers may be stacked to produce effective high-level features with characteristics like abstraction and invariance. Lower error reconstruction will be attained in this scenario, improving generalization expectations.

Using an LSTM-based autoencoder, the hybrid approach for anomaly identification in network traffic combines the benefits of statistical analysis with deep learning. With this connection, we want to improve our ability to spot abnormalities in sequential network traffic data. Various statistical characteristics are derived from the network traffic data during the data preparation stage. Important characteristics of the data, including mean, standard deviation, inter-arrival periods, and z-scores, are captured by these statistical features. A structured representation of the data is provided through statistical analysis, making it simple to evaluate and comprehend typical traffic behavior. The LSTM-based autoencoder is the deep learning model that will be used as part of the selected integration method. The statistical characteristics act as the autoencoder's encoder network's first input. The temporal relationships and sequential patterns in network traffic data are intended to be captured by the LSTM-based autoencoder. The encoder and the decoder are its two primary parts. The statistical information in the input is compressed by the encoder into a lower-dimensional representation known as the latent space. In order to reduce the reconstruction error during training, the decoder reconstructs the original input from the compressed representation. An instance-level labeled dataset of typical network traffic is used to train the hybrid approach. The autoencoder receives the statistical characteristics and the data's sequential nature as input. In order to ensure that it can precisely recreate the typical network traffic patterns using the statistical data, the autoencoder is trained to minimize the reconstruction error. Incoming network traffic examples are provided to the trained autoencoder for reconstruction during testing or real-time deployment. Low reconstruction errors are indicative of normal instances because they closely resemble the statistical features' statistical representations of acquired normal patterns. High reconstruction error instances show notable departures from expected behavior, warning of probable abnormalities. It is possible to categorize cases as normal or anomalous by setting a threshold for the reconstruction error. Statistical approaches or assessment metrics like ROC curve analysis can be used to calculate the threshold. Anomalies are defined as instances with reconstruction errors over the threshold, allowing the identification of network assaults or anomalous traffic patterns.

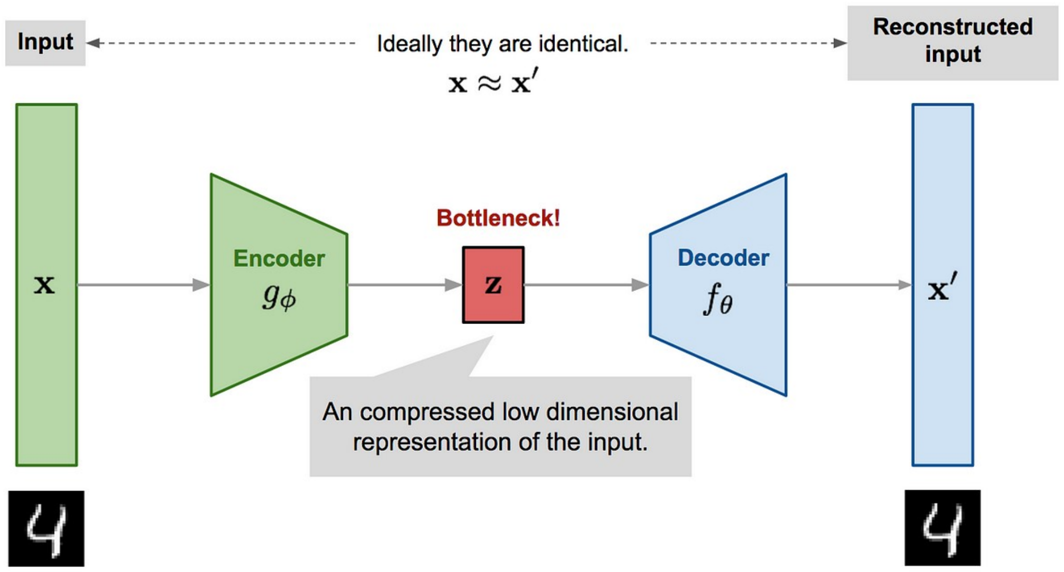

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Fig7. LSTM based autoencoder for anomaly detection

A training set and a testing set were created from the preprocessed dataset. The autoencoder was trained on the training set, and its performance on test data was assessed on the testing set The autoencoder received the preprocessed network traffic data as input, which was represented by statistical characteristics like z-scores and inter-arrival times. Encoder and decoder components made up the autoencoder architecture. The input data was compressed by the encoder into a lower-dimensional representation (latent space), and the decoder used this compressed form to recreate the original data. The Adam optimizer, a well-known stochastic optimisation technique that adjusts the learning rate for each parameter during training, was used to train the autoencoder. As for the loss function binary cross entropy is used. When working with models like logistic regression or neural networks with a sigmoid activation function in the output layer, binary cross-entropy loss (also known as log loss) is a frequently utilized loss function in binary classification issues. By calculating the difference between expected probabilities and actual binary labels, it evaluates the effectiveness of a binary classification model. When predicting one of two classes (e.g., spam or not spam, fraudulent or not fraudulent), it is utilized for binary classification problems. When predictions differ greatly from the actual labels, the loss function penalizes models more severely. False positives and false negatives are penalized differently by the loss because it is asymmetric.

$$L(y, p) = -[y * log(p) + (1 - y) * log(1 - p)] \quad .....(12)$$

where, $L(y, p)$ is the binary cross-entropy loss

y is the actual binary label (0 or 1)

p is the predicted probability of the instance belonging to class 1

A more accurate and sensitive anomaly detection system was created by the incorporation of the hybrid approach. The system may detect tiny irregularities and lessen false positives by combining the skills of statistical analysis and deep learning, improving network security. The hybrid method gave us a deeper comprehension of network traffic behavior. The deep learning model successfully caught long-term patterns and temporal relationships that are crucial for identifying complex network assaults, while the statistical analysis provided interpretable insights into the dataThe hybrid approach showed application in the Real World, making it a useful tool for network security operations. It is ideal for practical implementation in dynamic and varied network contexts because of the balance between accuracy and interpretability it exhibits.

In the future work of anomaly detection in network traffic, it involves continuous improvement and real-world deployment. The hybrid method showed real-world applicability, making it a useful tool for network administrators.Extensive Deployment Utilize the anomaly detection system in a live setting to keep track of actual network traffic. Think about the difficulties involved with managing a big amount of data, real-time processing, and system integration. Investigate the application of active learning approaches to include human analysts in the instruction process. Active learning allows for the manual selection of informative cases for labeling, allowing for human input to enhance model performance. functioning of security. It is ideal for practical implementation in dynamic and varied network contexts because of the balance between accuracy and interpretability it exhibits. Handling Deal with the issue of idea drift, in which network traffic patterns evolve over time. To make sure the model stays useful and current, devise techniques to identify and adjust for idea drift. Evaluate the system's performance in real-world settings while taking into account different network setups, traffic volumes, and network attack types. Evaluate how well it works in spotting both well-known and new oddities.

4. IMPLEMENTATION

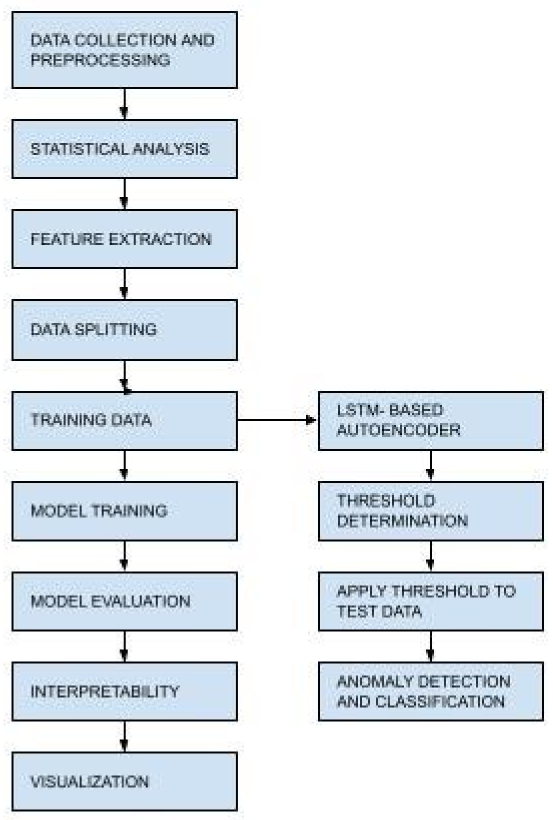
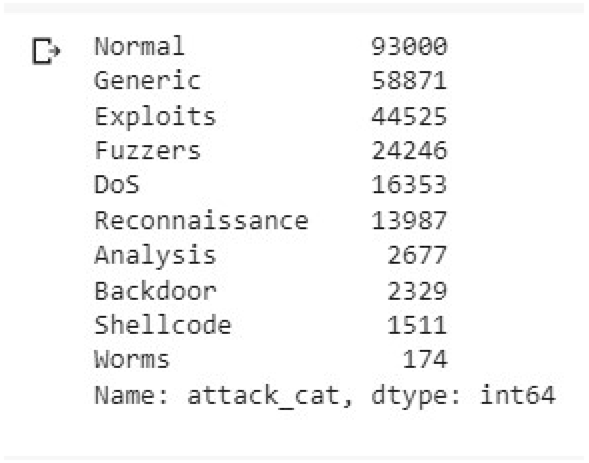


Fig8. Anomaly detection

4.1 DATASET

The data gathered is the UNSW-NB15 dataset from kaggle. The dataset contains regular network traffic and various types of attacks such as Denial of Service(DoS), port scanning and intrusion attempts. A total number of two million data is recorded in four csv files UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv and UNSW-NB15_4.csv. IXIA traffic generator used three virtual servers to compile these records.Two servers were set up to distribute regular network traffic, while a third was set up to produce irregular network traffic. Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and worms are among the nine attack categories in this dataset. To produce a total of 49 characteristics with the class label, the Argus and Bro-IDS tools are utilized, and 12 methods are built. To evaluate the effectiveness of models, the dataset is typically splitted into training and testing sets. The UNWS-NB15_training-set.csv and UNSW-NB15_testing-set.csv are partitions of this data with 175341 records in the training set and 82332 in the testing set.

Table 1 : Dataset record distribution



```
Normal            93000
Generic           58871
Exploits          44525
Fuzzers           24246
DoS               16353
Reconnaissance    13987
Analysis           2677
Backdoor           2329
Shellcode          1511
Worms               174
Name: attack_cat, dtype: int64
```

Table2: Features listed in the dataset

Attribute number	Attribute name	Attribute number	Attribute name	Attribute number	Attribute name	Attribute number	Attribute name
1	id	12	dttl	23	dtcpb	34	ct_dst_ltm
2	dur	13	sload	24	dwin	35	ct_src_dport_ltm
3	proto	14	dload	25	tcprtt	36	cst_sport_ltm
4	service	15	sloss	26	synack	37	ct_dst_src_ltm
5	state	16	dloss	27	ackdat	38	is_ftp_login
6	spkts	17	sinpkt	28	smean	39	ct_ftp_cmd

7	dpkts	18	dinpkt	29	dmean	40	ct_flw_http_mthd
8	sbytes	19	sjit	30	trans_depth	41	ct_src_ltm
9	dbytes	20	djit	31	response_body_len	42	ct_srv_dst
10	rate	21	swin	32	ct_srv_src	43	is_sm_ips_ports
11	sttl	22	stcpb	33	ct_state_ttl	44	attavk_cat
						45	label

4.2 DATA PREPROCESSING

A Python environment was used to import the UNSW-NB15 dataset using the Pandas module (version 1.3.3). A robust data manipulation package called Pandas offers data structures and operations to effectively manage huge datasets. It is a popular option for data analysis activities since it provides a variety of functions for data loading, data cleaning, and data preparation.In order to grasp the dataset's structure and spot any possible problems, we first looked through it.The training and testing sets are joined using the concatenation function. We learned important details about the data types and non-null counts for each feature via the ‘df.info()’ method. Preparing the data is essential to ensuring its quality and usefulness for subsequent analysis. Then used the df.isna().sum() function to handle any missing values, and it turned out that the dataset had no missing values. As a result, there is no longer a requirement for data removal or imputation because of missing data points. After that, df.drop(columns=['id', 'attack_cat']) is used to remove the 'id' and 'attack_cat' columns. For the investigation, the 'id' column functioned just as an identifier and did not include any useful data. The attack categories are represented by the 'attack_cat' column. It is removed from the dataset because it was a part of the test set and wasn't necessary for preprocessing. To represent the correlation between features in the dataset, a heatmap (sns.heatmap(df.corr())) created with Seaborn is used, a data visualization toolkit. As a result, we were able to pinpoint factors with significant correlations that could cause multicollinearity problems when modeling.

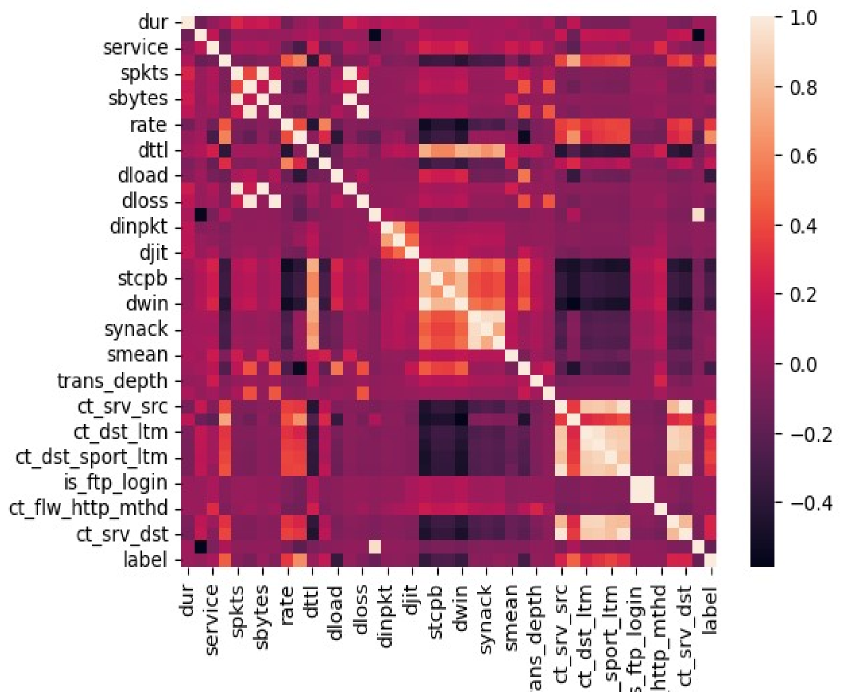


Fig9 : Heatmap showing correlation

The heatmap is a tool for visualizing pairwise correlations (relationships) between features in your dataset. It demonstrates the degree to which each characteristic is connected to every other feature. Each heatmap cell's color denotes the degree and slant (positive or negative) of the link. Stronger correlations are shown by darker colors (like various shades of red), whereas weaker or no connections are indicated by lighter colors (like various shades of blue). You can see which features are closely connected by looking at the heatmap. For instance, if two traits have a strong positive connection, they are likely to move in unison. On the other hand, if their negative association is strong, they are more likely to shift in the other way.

We kept the names of the highly correlated variables in the correlated_vars list to manage this multicollinearity. If necessary, during the model-building stage, additional actions can be done to handle these associated variables, such as eliminating one of the variables or using dimensionality reduction techniques. Using train_test_split from the sklearn package to divide the preprocessed data into training and testing sets in order to efficiently assess the performance of our models. The dataset was split into a training set and a testing set, with the training set having 70% of the samples and the testing set the remaining 30%. This allowed us to preserve a certain quantity of data for review while still training our models on a significant amount of data. The dataset is currently prepared for our research's latter phases, which will include feature engineering, model training, and assessment. The dataset's features and labels are extracted, and the features are scaled using the StandardScaler function of the scikit-learn library. Features and labels were separated out of the dataset. By deleting the 'label' column from the dataset, the features reflecting the analysis's input variables were retrieved. As the ground truth for the analysis, the target variable's value in the 'label' column was separated. To make sure that all features are on the same scale, feature scaling was then done. For many machine learning algorithms to function at their best, this phase is essential. For this, the StandardScaler from the scikit-learn module was used. It divides the standard deviation by the mean to scale the features, giving standardized features with a mean of 0 and a standard deviation of 1.

4.3 FEATURE EXTRACTION AND STATISTICAL ANALYSIS

This stage makes sure that the anomaly detection system makes use of the network traffic data's statistical properties and temporal relationships. Sequences are created from the network traffic data to input into the LSTM-based autoencoder. The data's temporal relationships are captured by the sequences, which enables the model to gradually learn the patterns. The relevant statistical traits discovered during the statistical analysis step are associated with each series of network traffic data. The sequential data and statistical characteristics are combined to provide the input for the LSTM-based autoencoder. The combined data is organized into the proper format needed for the LSTM-based autoencoder, which comprises both the sequential data and the statistical characteristics. Typically, this format uses 3D arrays or tensors, where the dimensions stand for the quantity of features, time steps, and sequences.

The DataFrame 'df''s features are obtained by using the formula features = df.drop('label, axis=1'). The target variable (the "label") is now separated from the input characteristics in this stage.The code then scales the features using StandardScaler from the scikit-learn module after extracting the features. The features are standardized using StandardScaler to have a mean of 0 and a standard deviation of 1.
Various approaches are employed based as shown in the Fig3 to the preprocessed data to obtain pertinent characteristics that characterize the behavior of network traffic during the

statistical analysis phase of anomaly detection in network traffic.The 'label' column is removed from the DataFrame 'df' in order to extract the features. Features in this context relate to the aspects or properties of network traffic data, including information like packet sizes, kinds of protocols, port numbers, and other pertinent details. Here some of the key statistical features used are (a) mean, the average value of a group of data points. It helps determine the normal traffic volume or activity level and sheds light on the data's fundamental tendency . (b) standard deviation, the spread or dispersion of data points around the mean is measured by the standard deviation. Greater variety in the traffic behavior is indicated by a higher standard deviation. The scaled_features array's features are divided into columns, and the first two lines of code calculate the mean and standard deviation of each column. (c) Standardized scores, or Z-scores, show how many standard deviations a data point deviates from the mean. Z-scores are helpful for locating data points that deviate unusually from the norm, possibly pointing to anomalies. Standardization is frequently carried out in order to compare results on the same scale.

$$x_i standardized = \frac{(x_i - \mu(x_i))}{\sigma(x_i)} \qquad \text{......(13)}$$

$$\text{Z-score} \qquad z_i = \frac{(x_i standardized - \mu(x_i standardized))}{\sigma(x_i standardized)} \qquad \text{......(14)}$$

x_i is the data point for the feature

$\mu(x_i)$ is the mean of that feature

$\sigma(x_i)$ is the standard deviation of that feature

$x_i standardized$ is the standardized value of the data point for a specific feature

$\mu(x_i standardized)$ is the mean of the standardized value

$\sigma(x_i standardized)$ is the standard deviation of the standardized value

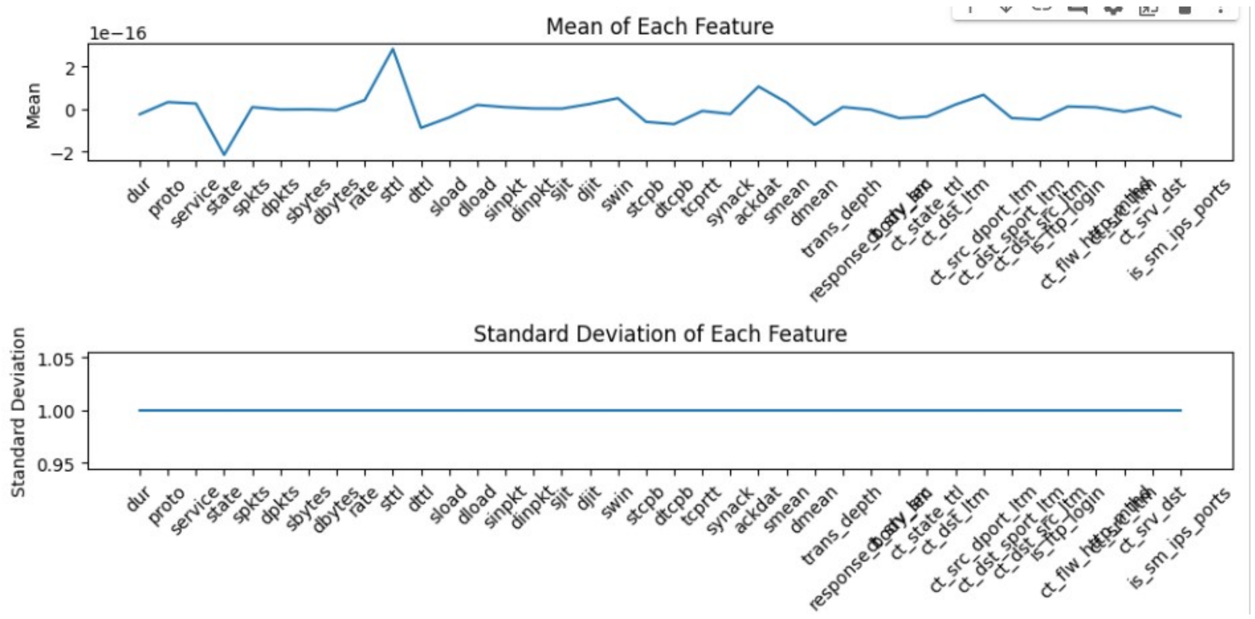


Fig10 : Representation of mean and standard deviation

Using Fig4. each feature's central tendency is represented by the mean. The average value for each characteristic is shown. To determine whether attributes have greater or lower average values, you may compare the means of the various features. Significant outliers in your data can affect the mean if there are any outliers at all. In this graph, outliers could show up as extreme numbers. The standard deviation calculates the range or variability of each feature's data points. A larger standard deviation shows that the distribution of the data points with respect to the mean is more scattered. Features with higher standard deviations may have wider data distributions, whereas features with lower standard deviations may have more concentrated data

(d) inter- arrival times, the periods of time between subsequent network packets or events. The timing of network operations may be analyzed to identify trends, which helps in the early discovery of anomalies. The differences between successive data points along each feature (column) in the z_scores array are computed using the np.diff() method.

$$IAT_i = z_i - z_{(i-1)} \qquad \textbf{........}(15)$$

z_i is the Z-score of the current data point

$z_{(i-1)}$ is the Z-score of the previous data point in the same feature

(e) traffic volume, It is possible to spot abrupt increases or decreases in network activity, which could be signs of unusual occurrences, by analyzing the overall volume of traffic over a period of time. To get the number of rows, which equates to the number of packets in the network traffic dataset, it uses the.shape parameter of the DataFrame 'df'.(f) Examining the distribution of network traffic among several protocols reveals instances of atypical protocol usage that could point to security flaws. Here it figures out how the packets' various protocols are distributed. Using the value_counts() function, it determines the number of times each distinct value appears in the 'proto' column of the DataFrame. (g) It is possible to identify odd port scanning or unusual service use, which are frequent signs of network assaults, by analyzing the distribution of traffic across several ports.. Using the value_counts() function, it determines the number of times each distinct value appears in the 'proto' column of the DataFrame. This gives information on the most frequently utilized protocols in the network traffic. (h) The distribution of packet sizes can be examined to find differences that might be a sign of certain network assaults or abnormalities. The value_counts() function is used to count the instances of each distinct packet size value in the DataFrame's'spkts' column. It may get a sense of the distribution of packet sizes in the network traffic from this.

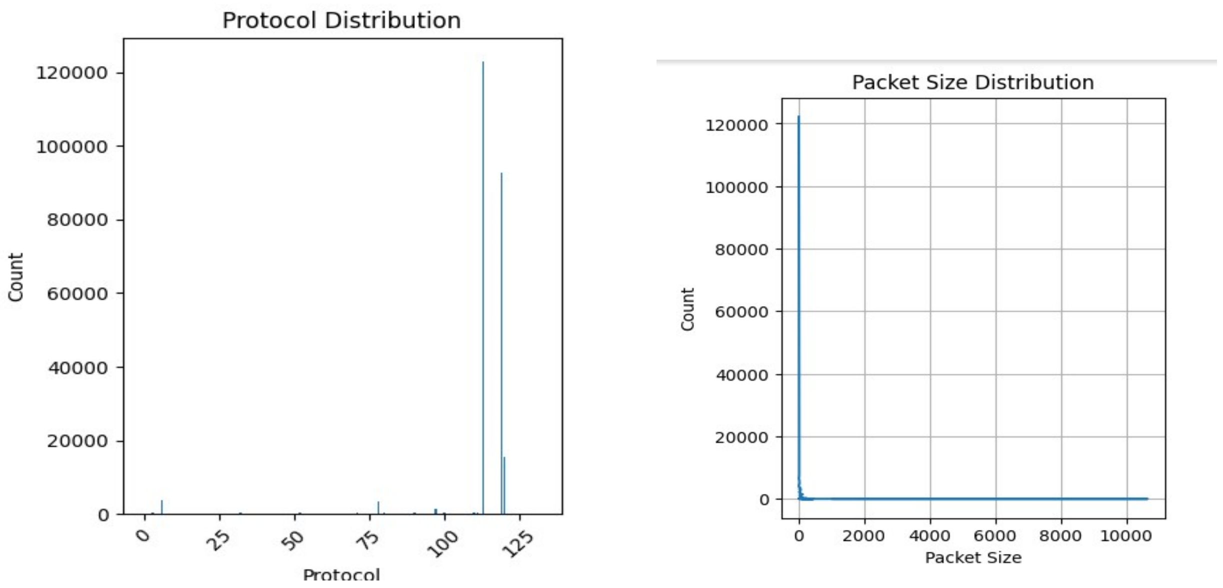


Fig11 : protocol and packet size distribution

The first graph in Fig11: makes it easy to see which network protocols are utilized the most. The procedures with the highest bars are usually the dominant ones. Understanding the prevalent network activity kinds is made easier by identifying these dominating protocols. Unusual network behavior or abrupt changes in the distribution of a certain protocol might be an indication of an anomaly. Unusual distributions of protocols may point to security problems. For instance, a sudden increase in traffic for a less popular protocol may indicate malicious activity. It assists in resource optimisation for the network. It may be required to implement Quality of Service (QoS) regulations or more effectively distribute resources if some protocols use a significant amount of bandwidth.

The line graph in Fig11: illustrates the distribution of packet sizes. It aids in comprehending the common patterns of packet size in your network traffic. Packet sizes for various applications and network activity may vary. Differences from the typical packet size distribution may be a sign of trouble. A rapid rise in packet sizes that are excessively huge, for instance, might indicate a network assault. Examining packet sizes can reveal information about the efficiency of a network. It aids in comprehending the efficiency of data transport as well as the overhead caused by packet headers. By modifying Maximum Transmission Unit (MTU) settings or employing compression methods for particular types of traffic, one may increase network bandwidth by having a better understanding of packet size distributions.

The system may extract useful characteristics that accurately depict typical network behavior by using these statistical analysis approaches on the preprocessed network traffic data. With the help of these statistical characteristics, the LSTM-based autoencoder will be better able to identify pertinent patterns and deviations in network data, producing results for anomaly detection that are precise and easy to understand.

4.4 MODEL TRAINING

The LSTM-based autoencoder is taught using the training set during the model training phase so that it may learn to encode and reconstruct typical traffic patterns. By choosing hyperparameters like the number of LSTM layers, hidden units, and learning rate, the architecture of the autoencoder is determined. The model's performance is optimized using the validation set, and it is confirmed that these hyperparameters can precisely capture and recreate typical network traffic patterns.

The encoder and the decoder are the two primary components of the autoencoder. The decoder reconstructs the original input from the compressed form after the encoder has compressed the

input data into a lower-dimensional representation. By contrasting the reconstructed output with the actual input, anomalies are found.
An LSTM layer with 128 units and the hyperbolic tangent activation function serve as the foundation of the encoder. This layer accepts input that has a form that matches the time steps and characteristics of the training data. To avoid overfitting, a dropout layer is added. The output is duplicated by the RepeatVector layer to take the form of the original input.

Multiple LSTM layers are used in the decoder stage to gradually recover the original input. Decreasing units (64, 32, and 16) are used in these levels to gradually extend the compressed representation. A dropout layer is placed after each LSTM layer to prevent overfitting. The final layer reconstructs a single feature point and is a fully linked dense layer with a sigmoid activation function.

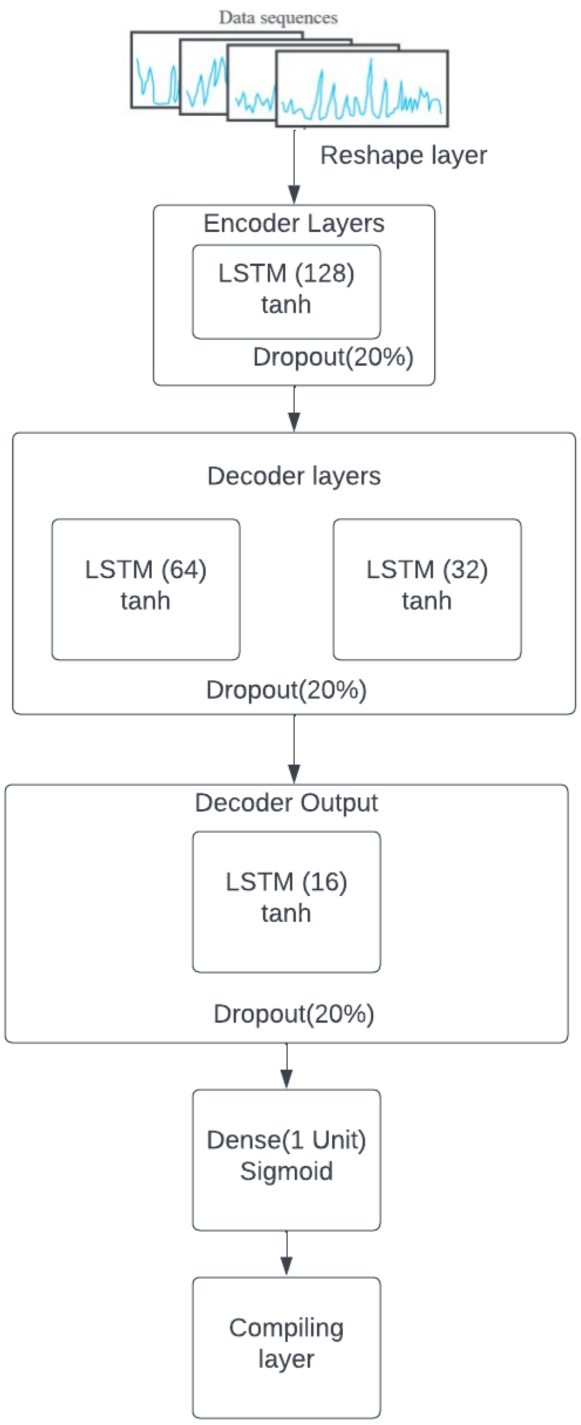


Fig12: LSTM - Autoencoder Model

The 'adam' optimizer and the 'binary_crossentropy' loss function are used in the model's compilation. We seek to reduce the cross-entropy loss between the reconstructed outputs and the original inputs because this is an anomaly detection challenge.

The data is split into batches of size 128 in order to train the model. There are 5 epochs in the training process. In addition, a validation split of 20% is applied to track the model's performance on training data that has not yet been viewed. To guarantee that the data's temporal order is preserved during training, the parameter shuffle is set to False.

Table 3:Proposed LSTM-Autoencoder model

Layer	Type	Filters/Neurons	Activation
1	Input(LSTM)	128	tanh
2	Dropout	-	-
3	Repeat vector	-	-
4	LSTM	64	tanh
5	Dropout	-	-
6	LSTM	32	tanh
7	Dropout	-	-
8	LSTM	16	tanh
9	Dropout	-	-
10	Dense(Output)1		sigmoid

By calculating the reconstruction loss between the original data and the rebuilt output after the model has been trained, anomalies may be located. Anomalies may be considered to be cases with higher reconstruction loss values that depart from learnt patterns.

The LSTM-based autoencoder must learn the representations of typical network traffic patterns during the model training phase. During the testing phase, the model's performance is optimized with the aid of the validation set's hyperparameter fine-tuning, making it more capable of spotting abnormalities in real-world network traffic.

4.5 Model Evaluation

The evaluate() function computes preset metrics on the test dataset in order to evaluate the model. The following are often utilized critical assessment criteria for this anomaly detection task:

Loss Value (Cross-Entropy) measures how different the projected results are from the actual labels. A larger loss value for anomaly identification frequently denotes a bigger departure from typical patterns.

While accuracy is frequently employed for classification tasks, due to the unbalanced nature of anomaly data, it may be less relevant for anomaly detection. The capacity of the model to accurately identify abnormalities is more important.

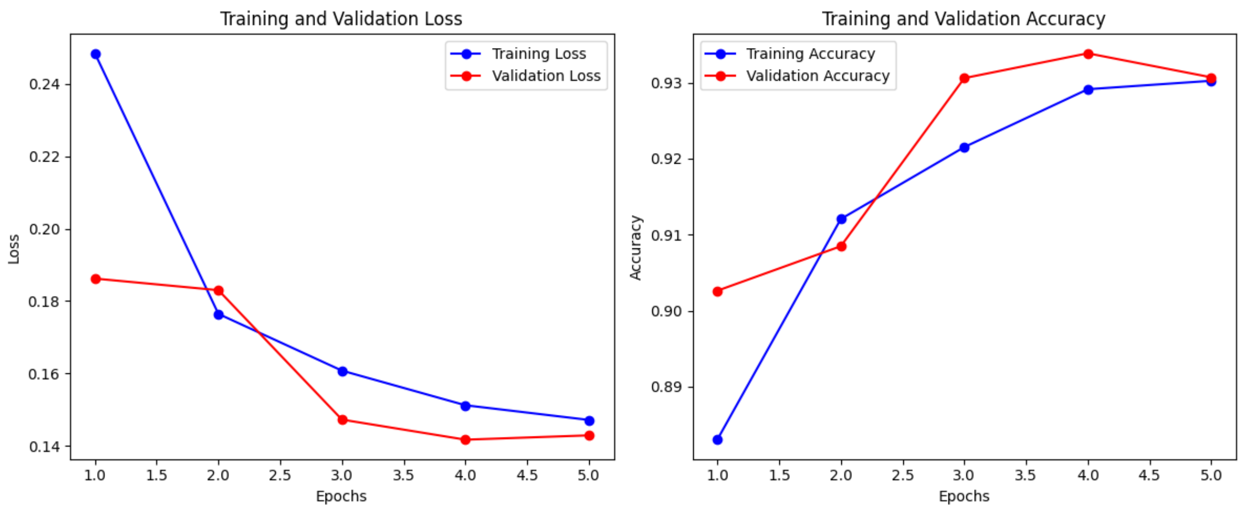


Fig 13: Loss and accuracy per epochs

In the Fig13 we can see the model is adapting and generalizing well if the training and validation loss are both falling and convergent. With overfitting, the model may be fitting the training data too closely and failing to generalize to new data if the training loss declines but the validation loss increases or diverges. Want to see both training and validation accuracy rising and convergent, similar to loss. Overfitting may be revealed by a wide discrepancy between training and validation accuracy. If the model performs well during training but poorly during validation, it may be because it has trouble remembering the training set of data.

5.RESULTS

5.1 EVALUATION

The LSTM- based autoencoder model’s evaluation results on the test data are

Table4 :Results

Accuracy	**93.02%**
Val Accuracy	**93.07%**
Loss	**0.1472**
Val Loss	**0.1429**

The determined loss value of 0.1472 offers crucial information about the model's effectiveness. The difference between the model's predictions and the actual target values is quantified by the loss value in the context of this anomaly detection job. In other words, a smaller loss value suggests that the model successfully captures typical patterns since the model's reconstructions closely match the input data.

The claimed accuracy of 93.02% provides a basic sense of how well the model performs in categorizing data. However, because of the data's imbalance, accuracy in jobs involving anomaly detection can be a little deceptive. The model is biased towards normal situations since anomalies are often far more uncommon than regular occurrences. As a consequence, even if the model has trouble properly identifying abnormalities, a high accuracy may still be attained.

While the loss value and accuracy offer insightful data, a thorough evaluation of the model's performance takes into account other elements. The capacity of the model to reliably detect anomalies (high true positive rate) while minimizing false positives and false negatives is frequently prioritized in jobs for anomaly detection. To acquire a more detailed knowledge of the behavior of the model, it is crucial to analyze measures like accuracy.

5.2 RELATED WORKS

Table5 :Summary of related anomaly detection strategies

	Category	**Method**	**Dataset**	**Performance**	**Year**
[1]	Anomaly detection	C-LSTM	Yahoo webscope s5	98.60%	2018
[2]	Anomaly detection	CNN and Autoencoder	USTC-TFC2016	100%	2020
[3]	Anomaly detection	LSTM and AM	CSE-CIC-IDS2018	96.2%	2019
[4]	Anomaly detection	LWPR	Wireless sensor network	86%	2019
[9]	Intrusion Detection	Machine learning	NSL-KDD	98.3% (Young and Bae)	2020
[12]	Anomaly detection	SVM-L	Self collected	99%	2021
[13]	Intrusion Detection	Recursive Feature Elimination and Random Forests	UNSW-NB15	74%	2019

Here, we can observe that the C-LSTM's accuracy of 98.60% is better. because using a technique with 100% accuracy may lead to overfitting. Additionally, the UNSW-NB15 training and testing dataset was the one I selected. With the dataset, intrusion detection accuracy is 74%. However, the accuracy of the UNSW-NB15 data using the LSTM-based autoencoder was 93.02%.

5.3 CHALLENGES AND SOLUTIONS

There might be a number of difficulties while implementing the anomaly detection system utilizing the hybrid strategy of statistical analysis and LSTM-based autoencoder. The effectiveness and performance of the system may be impacted by these problems.

Overfitting:
The difficulty comes from overfitting, which happens when a model does well on training data but badly on unobserved data.
Solution: Use strategies like regularization or dropout to stop overfitting. Additionally, when validation performance begins to deteriorate, employ early stopping to end training.

Anomaly detection jobs frequently have unbalanced class distributions, with anomalies being more uncommon than regular occurrences.
Solution: To solve class imbalance and avoid the model from becoming biased towards the dominant class, use approaches like oversampling, undersampling, or class weighting during training.

Data Preprocessing Difficulties
Challenge: It might be difficult to make sure that the data is correctly preprocessed and prepared for the hybrid technique.
Follow industry standards for data pretreatment, normalization, and formatting as a solution. Use libraries like Scikit-learn or TensorFlow to ensure consistency and correctness.
By addressing these challenges through appropriate techniques and strategies, it can enhance the robustness, efficiency, and accuracy of your anomaly detection system.

The hybrid anomaly detection system's successful deployment represents a substantial improvement in network security. The system enables precise and understandable anomaly identification in network data by combining the benefits of statistical analysis with LSTM-based autoencoders. This strategy lessens the load of false positives while enabling the system to accurately detect even the most minute irregularities. Deep learning and statistical analysis together have a special set of advantages for network security.

The drive to improve network security will continue after this implementation, though. Future research will concentrate on enhancing the system's performance by utilizing more statistical methods and investigating more complex deep learning architectures. Given how quickly the threat landscape changes, ongoing adaptation is essential. The system will continue to be on the lookout for new network threats, adjust its tactics as necessary, and provide the highest degree of security.

In conclusion, it gives organizations a strong instrument to defend their networks against a variety of possible attacks because of its demonstrated effectiveness, interpretability, and flexibility. The anomaly detection system will develop along with technology, persistently protecting networks and safeguarding the integrity of digital environments.

6. CONCLUSION

The practical value of the model's outputs is also greatly influenced by domain expertise and business requirements. Depending on the application environment, false positives and false negatives might have different effects, misclassifying regular cases as anomalies or failing to find real abnormalities.

Our main goal has been to build and install an efficient anomaly detection system that can spot suspicious activity in network data over the whole lifespan of the project. To provide a solid solution, we made use of machine learning approaches and techniques LSTM-autoencoder. For the purpose of ensuring the viability and efficacy of our system, its functionality and usability have undergone a thorough evaluation.

6.1 REFLECTION

The project's ability to achieve its aims may be credited to its well defined objectives. Each project phase was guided by clearly defined aims and objectives, which ensured concentration and alignment with the desired results. The performance of the model may be evaluated using precision, recall, F1-score, and area under the ROC curve. Based on the ROC curve, we may modify the anomaly detection threshold to strike a balance between false positives and false negatives. Working together with mentors, peers, and subject matter experts was essential to solving problems and coming to wise judgements. Getting other people's opinions and ideas improved the project's quality. It was essential to be able to adjust to shifting conditions and project needs. Problem-solving was more productive when project plans and techniques could be modified as needed.

6.2 FUTURE WORKS

Although a strong basis for anomaly detection in network data has been established by this research, there are still a number of areas that may be explored and improved. The installation of the anomaly detection system on a real-time, online platform is one of the next stages that should be taken right away. This would include integrating the model into a functioning network architecture, monitoring incoming traffic constantly, and sending out prompt notifications whenever abnormalities are found. For a quick reaction to network threats, real-time deployment is essential. Making sure the anomaly detection system can scale as network traffic data quantities increase is crucial. The performance of the system should be optimized in the future to effectively manage large-scale network settings. Add interactive data visualization and reporting options to improve the user interface. This helps decision-making and gives network managers broader understanding of network abnormalities. Join forces with organizations, researchers, and specialists in the subject of cybersecurity to share expertise, verify theories, and advance the body of knowledge in the area.

7. RESEARCH PAPERS

1. Young, Kim Teang, and Cho SSung Bae. “Web traffic anomaly detection using C-LSTM neural networks.” *Web traffic anomaly detection using C-LSTM neural networks*, 2018, https://www.sciencedirect.com/science/article/abs/pii/S0957417418302288. Accessed 15 August 2023.

2. R. -H. Hwang, M. -C. Peng, C. -W. Huang, P. -C. Lin and V. -L. Nguyen, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," in IEEE Access, vol. 8, pp. 30387-30399, 2020, doi: 10.1109/ACCESS.2020.2973023.

3. Lin, P., Ye, K., Xu, CZ. (2019). Dynamic Network Anomaly Detection System by Using Deep Learning Techniques. In: Da Silva, D., Wang, Q., Zhang, LJ. (eds) Cloud Computing – CLOUD 2019. CLOUD 2019. Lecture Notes in Computer Science(), vol 11513. Springer, Cham. https://doi.org/10.1007/978-3-030-23502-4_12

4. Poornima, I. Gethzi Ahila, and B. Paramasivan. "Anomaly detection in wireless sensor networks using machine learning algorithms." *Computer communications* 151 (2020): 331-337.

5. Komisarek, Mikolaj, et al. "Machine Learning Based Approach to Anomaly and Cyber Attack Detection in Streamed Network Traffic Data." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 12.1 (2021): 3-19.

6. Rao, K. Hanumantha, et al. "Implementation of anomaly detection techniques using machine learning algorithms." *International journal of computer science and telecommunications* 2.3 (2011): 25-31.

7. Ma, Chencheng, Xuehui Du, and Lifeng Cao. "Analysis of multi-types of flow features based on hybrid neural networks for improving network anomaly detection." *IEEE Access* 7 (2019): 148363-148380.

8. Hareesh, I., et al. "Anomaly detection system based on analysis of packet header and payload histograms." *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*. IEEE, 2011.

9. Vikram, Aditya. "Anomaly detection in network traffic using an unsupervised machine learning approach." *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2020.

10. Hu, Fei. *Security and privacy in Internet of things (IoTs): Models, Algorithms, and Implementations*. CRC Press, 2016.

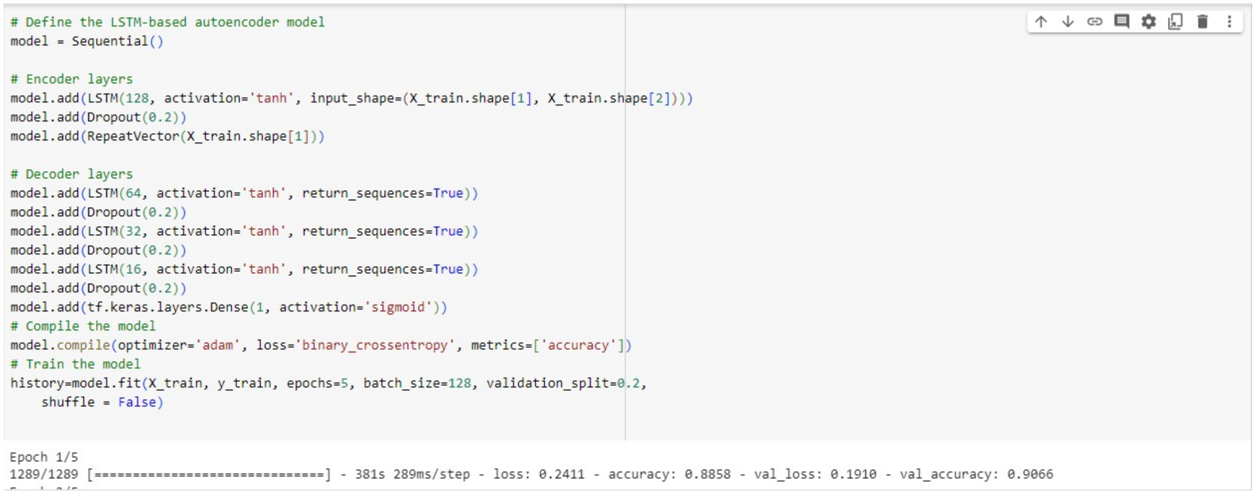
11. Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015.

12. Ma, Qian, et al. "A novel model for anomaly detection in network traffic based on kernel support vector machines." *Computers & Security* 104 (2021): 102215.

13. Meftah, Souhail, Tajjeeddine Rachidi, and Nasser Assem. "Network based intrusion detection using the UNSW-NB15 dataset." *International Journal of Computing and Digital Systems* 8.5 (2019): 478-487.

8.APPENDIX

1. Model building and training



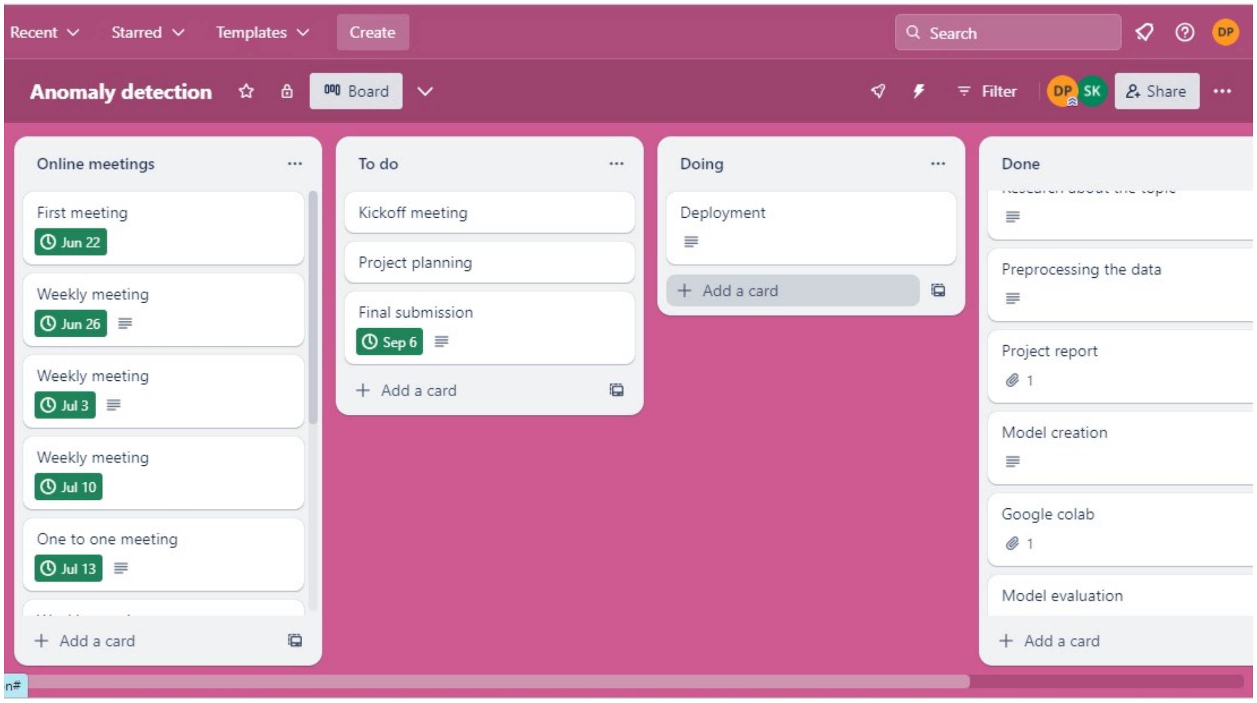
```
# Define the LSTM-based autoencoder model
model = Sequential()

# Encoder layers
model.add(LSTM(128, activation='tanh', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(RepeatVector(X_train.shape[1]))

# Decoder layers
model.add(LSTM(64, activation='tanh', return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(32, activation='tanh', return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(16, activation='tanh', return_sequences=True))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
history=model.fit(X_train, y_train, epochs=5, batch_size=128, validation_split=0.2,
    shuffle = False)
```

```
Epoch 1/5
1289/1289 [==============================] - 381s 289ms/step - loss: 0.2411 - accuracy: 0.8858 - val_loss: 0.1910 - val_accuracy: 0.9066
```

2. Project management tool



Online meetings
- First meeting — Jun 22
- Weekly meeting — Jun 26
- Weekly meeting — Jul 3
- Weekly meeting — Jul 10
- One to one meeting — Jul 13
- Add a card

To do
- Kickoff meeting
- Project planning
- Final submission — Sep 6
- Add a card

Doing
- Deployment
- Add a card

Done
- Research about the topic
- Preprocessing the data
- Project report — 1
- Model creation
- Google colab — 1
- Model evaluation
- Add a card

3. Project delivery Schedule

Note: Reorder the sections in the order that you plan to complete them.	
Abstract	28/08/2023
Declaration	28/08/2023
Acknowledgements	28/08/2023
Introduction	20/07/2023
Literature - Technology Review	30/07/2023
Methodology	28/07/2023
Implementation and Results · Evaluation · Related Work	01/08/2023
Conclusion · Reflection · Future Work	25/08/2023
References	25/08/2023
Appendices	28/08/2023

4. Artefact Delivery schedule

Note: Reorder the activities in the order that you plan to complete them.	Deadline Date

Artefact Planning and Resourcing	20/07/2023
Artefact Design	25/07/2023
Artefact Procurement Activities (e.g., data collection, source framework etc.)	31/07/2023
Artefact Development, Deployment, Implementation	15/08/2023
Artefact Evaluation and Testing	25/08/2023
Artefact Presentation and Demonstration	06/09/2023
Artefact Screencast	08/09/2023

5. **Trello login Link**
6. **Github Link**