National College of Ireland

Intrusion Detection in Cloud Environments using Hybrid Deep Learning

MSc Research Project
MSc in Cloud Computing

Kunal Rana
Student ID: 22141138

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Kunal…Rana……

Student ID: …22141138……

Programme: …MSc…in…Cloud…Computing…… **Year:** ……2023……..

Module: …MSc…Research…Project……

Supervisor: …Shaguna…Gupta……

Submission Due Date: ……

Project Title: … Intrusion Detection in Cloud Environments using Hybrid Deep Learning ……

Word Count: ……25…… **Page Count**……10023……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Kunal…Rana……

Date: ……25/04/2024……

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):	

Intrusion Detection in Cloud Environments using Hybrid Deep Learning

Kunal Rana
22141138

Abstract

In todays mordern word ensuring strong security measures against network intrusions is essential for cloud computing. This study investigates the use of deep learning and machine learning approaches for intrusion detection to improve cloud environment security. It looks at how well different models—such as Bidirectional LSTM, Decision Trees, Long-Short Term Memory (LSTM), and Logistic Regression—identify and categorize network intrusions. These models are assessed using accuracy, recall, and F1 score metrics on the UNSW-NB15 dataset. The study also explores how to pick features and how to employ autoencoders to improve model performance. The outcomes provide important insights into the potential real-world applications of each algorithm by highlighting its advantages and disadvantages.

1 Introduction

This thesis addresses the growing threat of intrusions and presents a novel solution by integrating machine learning and data learning methods into cloud security. The socially engineered techniques used in various intrusion types, such as DoS, backdoor, shellcode, etc., are frequently too strong for conventional defense measures to handle. The study uses the extensive dataset known as (UNSW-NB15) to address this problem. It has Fuzzers, analysis, backdoors, denial-of-service, exploits, generic, reconnaissance, shellcode, and worms are the nine types of attacks included in this dataset. In order to create a total of 49 features with the class label, twelve algorithms are constructed, and the Argus and Bro-IDS tools are utilized. Cloud computing has completely changed the IT environment because it provides unmatched scalability, accessibility, and efficiency. Effectively managing and transferring data across large networks is a key component of this change. However, a variety of security threats are rapidly posing a threat to the vital assets housed in cloud technologies (Moustafa, 2015).

With distributed content delivery networks (CDNs), cloud service providers like AWS, Azure, and GCP have made it easier for data to be shared globally while offering quick content delivery through localized edge points. Although these developments improve performance and accessibility, they also expose data to increased hazards associated with long-distance network traffic. As a result, network-based assaults that target cloud settings are now a serious threat.

For network security defenses to be strengthened, it is essential to comprehend and counteract routing-based threats. A wide range of advanced attacks, from traditional Denial of Service (DoS) attacks to sneaky penetration methods like SQL injection and malware distribution, are included in the spectrum of potential threats. The confidentiality, availability, and integrity of cloud resources are by these vulnerabilities (Zekri, 2017).

Furthermore, to protect cloud infrastructures against newly discovered vulnerabilities, ongoing research, and innovation are required due to the dynamic nature of cyber threats. Even while cloud computing has the potential to revolutionize many industries, its broad adoption depends on strong security protocols that can mitigate a variety of dangers. This article includes relevant work Hybrid deep learning, a relatively new and creative method for intrusion detection in cloud environments, is explained in Section 4 along with the deep learning models used for machine learning. It has the ability to drastically lower different threats by using algorithms and data analysis to identify and stop invasions. The use of machine learning and deep learning methods for intrusion detection using cloud infrastructure is the main focus of this research. In Section 3, the research technique is explained. The design elements of the framework for machine learning algorithms are covered in Section 5. Section 6 details how this study was carried out. The evaluation results are presented and discussed in Section 7. The study is concluded in Section 8 with a discussion on future work. has context menu.

2 Related Work

The literature review presented in this paper offers a comprehensive overview of recent findings and methods related to intrusion detection systems (IDS) in cloud computing environments. Efficient security solutions that are optimized for cloud environments are essential due to the rapid use of cloud services and a growing number of cyber threats. A variety of studies with different viewpoints on the creation, use, and evaluation of intrusion detection systems are included in the review. By examining recent advancements in machine learning, deep learning, and hybrid approaches, the literature review highlights the importance of strong security frameworks in safeguarding cloud infrastructures and demonstrates how intrusion detection is evolving. The review offers a foundation for the methodology employed in this work as well. (Alqahtani, 2020)

2.1 Comparative Analysis of Deep Learning Techniques and Dataset Evaluation in Network Intrusion Detection

An in-depth analysis of deep learning methods for intrusion detection is presented by Ali Azawii, with a focus on widely used datasets (DARPA, KDD99, NSL-KDD), as well as deep learning frameworks (TensorFlow, Theano, and Caffe). In-depth discussions of autoencoders, sum-product networks, and recurrent neural networks among the deep learning architectures are provided in this publication. Along with suggestions for further study, including the use of hybrid approaches and the mixing of algorithms to improve performance, it emphasizes the significance of feature extraction and selection in making improvements in accuracy (Azawii, 2019). On the other hand, a research paper by Nour Moustafa and Jill Slay compares the effectiveness of two popular datasets for network intrusion detection system (NIDS) evaluation: KDD99 and UNSW-NB15. It suggests a thorough technique that is divided into three layers: the network layer, the processing layer (using decision engines and Association Rule Mining), and the assessment layer for NIDS effectiveness assessment. In summary, the report indicates that UNSW-NB15 features outperform KDD99 in many circumstances, but it also highlights the need for more research in this area by noting how difficult it may be to discern between comparable record values in datasets. Both publications

provide significant contributions to the field of intrusion detection; Nour Moustafa and Jill Slay focus on dataset comparison and NIDS assessment methodology, while Ali Azawii concentrates on deep learning frameworks and methodologies (Moustafa, 2015).

2.2 Innovations in Cloud-Based Intrusion Detection Systems

Both M. Mayuranathan and RM Balajee, the authors of the two articles, offer creative methods for intrusion detection in cloud systems. A hybrid approach that combines deep learning algorithms, clustering, optimization, and dimensionality reduction is proposed by RM Balajee to address the dimensionality and performance issues in attack detection. Through the integration of both traditional and deep learning approaches, their approach offers a comprehensive solution for faster and more accurate attack detection. AutoEncoder, FCM, PCA, and SMO are integrated (Balajee, 2023). On the other hand, intrusion detection is handled by the deep Kronecker neural network (DKNN) in the EOS-IDS model, which is tailored for cloud computing environments. In order to enhance detection accuracy, this model prioritizes feature selection, optimal pre-processing, and hybrid deep learning methods. (M. Mayuranathan, 2022).

2.3 Hybrid Intrusion Detection Systems in Cloud Environments: A Comparative Analysis

This article presents a hybrid intrusion detection system (IDS) by Ammar Aldallal and Faisal Alisa that combines genetic algorithms (GA) for feature selection with support vector machines (SVM) for classification. The GA is used to optimize feature selection by reducing the dataset's dimensionality and raising the efficiency of SVM classification. Numerous SVM kernel functions are examined; the most effective ones are discovered to be polynomial and linear kernels, especially when the feature set is reduced to 20, or one-fourth of the original features. The impressive results of 100% accuracy obtained with a minimal number of features show how effective this strategy is; nonetheless, further investigation into various machine learning (ML) techniques is recommended to confirm. In order to improve detection accuracy, the system uses PCA for dimensionality reduction in conjunction with data preparation techniques including traffic analysis and categorization. While SVM is utilized for anomaly detection, K-means clustering is employed for data categorization. The system exhibits scalability for integrating many virtual machines and increasing analysis to include VM IDs, with an emphasis on contextual analysis and the relationship between network and application logs. When compared to standalone IDSs, it also offers better accuracy and quicker detection times. (Sarosh, 2021).

2.4 Contrasting Approaches to DDoS Attack Detection in Cloud Environments

The important issue of identifying Distributed Denial of Service (DDoS) attacks in cloud computing environments is addressed in both research studies, albeit with different methods and strategies. In addition to several intrusion detection techniques, including Signature-based Detection (SD), Anomaly-based Detection (AD), and Stateful Protocol Analysis, the study by Marwane Zekri examines machine learning techniques including Naive Bayes, Decision Trees (C4.5), and Neural Networks (SPA). The study addresses the importance of

DDoS detection systems for cloud platform security and offers suggestions for further research on real-time attack traffic detection and mitigation (Zekri, 2017). However, the research by Aanshi Bhardwaj presents a DDoS attack detection method that employs Deep Neural Networks (DNN) optimized by the Ant Colony Method (ACO). In this 4-phase study, the CICIDS2017 dataset is used for pre-processing, pre-training, training, and testing; the results indicate good performance in detecting DDoS traffic in cloud environments. A temporal complexity study of deep learning approaches for DDoS detection and more real-world validation are two possible directions for future research that are highlighted in the paper's conclusion. In conclusion, both studies provide important viewpoints and approaches for addressing the security threats put on by DDoS attacks in cloud computing environments. (Bhardwaj, 2021).

2.5 Comparative Analysis of Data-Driven and Deep Learning Approaches for Intrusion Detection Systems

These research publications revolve around the development of machine learning-based intrusion detection systems (IDS); deep learning may be emphasized in different ways. Hamed Alqahtani uses the KDD'99 Cup dataset and other traditional machine-learning techniques to model IDS. The research highlights the significance of comprehending the nature of cyber-security data and extracting pertinent features through its comprehensive approach to dataset preparation and examination. Classification issues are addressed by a variety of machine learning techniques, including Decision Trees, Random Forests, Artificial Neural Networks, Bayesian Networks, and Naive Bayes. The evaluation of the success of the IDS model focuses mostly on performance indicators, such as accuracy, precision, recall, and F-score. The study concludes that a data-driven strategy is essential for providing intelligent cyber-security services (Alqahtani, 2020). However, Santhosh Parampottupadam is more concerned in using deep learning models to detect network intrusions in real time. The work starts with an awareness of the problem and an assessment of the literature, using the CRISP-DM methodology and highlighting the need to investigate the possibilities of deep learning in this field. Because the NSL-KDD dataset is of greater quality than KDDcup99, it was used. In particular, binomial and multinomial deep learning models are being created to predict intrusions and classify attacks, respectively. When comparing H2O deep learning models with other machine learning techniques, evaluation metrics like accuracy and detection rates show better results. (Moldovann, 2018).

2.6 Comparative Analysis of Cloud Security Frameworks: Traditional vs. Machine Learning Approach

An comprehensive security framework for efficiently detecting network breaches is provided in a research paper written by Rajendra Patil. Scalability, accuracy, and a low number of false alarms are the main priorities of the method, which makes use of techniques including multi-threaded models, Random Forest classifiers, signature-based detection, and feature selection using an extended Bat Algorithm. Comprehensive recommendations for future work are made, and quick processing times and good detection rates are demonstrated by experimental validation (Rajendra Patil, 2019). On the other hand, a work written by Geetika Tiwari presents an improved cloud security intrusion detection system based on machine learning and deep learning. The process it describes involves gathering data, preparing it, selecting

features, and categorizing it in multiple ways. As evidenced by the experimental results, the proposed model is effective in improving attack detection abilities and potentially reaching up to 97% classification accuracy. Possible future research directions include improving categorization efficiency and creating techniques for packet profiling. As a result, whereas Rajendra Patil focuses on a comprehensive security architecture that emphasizes efficiency and scalability, Geetika Tiwari's study offers a machine learning and deep learning-based strategy with improved threat detection skills and the potential for high accuracy. These papers provide useful advice on enhancing cloud security through the application of cutting-edge intrusion detection methods. (Jain, 2022).

2.7 Comparative Analysis: Diverse Approaches to Intrusion Detection in Cloud Security

This paper is by Muhammad Salman Saeed where the author discusses the nature and mitigation of distributed denial-of-service (DDoS) attacks that are a significant threat to cloud systems. The article stresses the lack of efficient intrusion detection systems and provides the reader with a detailed overview of the several kinds of DDoS attacks and their causes. In digging into the screening of intrusion detection, this work particularly pays attention to three methods, namely: signature-based detection, anomaly-based detection, and stateful protocol analysis. Moreover, the research also focuses on the effectiveness of machine learning techniques namely as Random Forest, Naive Bayes, SVM, and decision trees for the identification of intrusions. Based on experience, it is possible to conclude that the proposed work contributes significantly to using the Random Forest classifier for improving the feature selection and enhancing the accuracy of the intrusion detecting model up to 97,5% (M. S. Saeed, 2022). On the other hand, K. Shanthi's paper discusses basically the Anomaly based IDS and focuses more on the use of machine learning techniques for accurate anomalies detection. This technique entails applying SVM for this anomaly detection and for isolation forest model. While classification is used by SVM to label anomalies the isolation forest model on the other hand uses recursive partitioning to isolate the peculiarities of network traffic data. Categorization is performed on input data after it has gone through feature extraction methods such as auto-encoder structures. The proposed isolation forest model is then compared with the SVM model on the NSL-KDD dataset and it is observed that the SVM model performs slightly better than the isolation forest model. Consequently, the anomaly-based machine learning models were seen to offer very fast and accurate outcomes in terms of anomaly detection and especially when working with very big datasets that consume very little memory. (Maruthi, 2023).

Index	Research Papers	Release Date	Authors	Results	Limitations
1	Survey on Intrusion Detection Systems based on Deep Learning	2019	Ali Azawii, Sufyan T. Faraj Al-Janabi, Belal Al-Khateeb	High precision and detection rate which reached approximately 99%.	Limitations include dataset limitations, a framework that focuses on TensorFlow, and a lack of architecture reviews. Model interpretability, complexity, and overfitting are all ongoing challenges. Resolving these enhances future research.
2	The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems	2015	Moustafa, Slay	The UNSW-NB15 dataset demonstrates that the suggested feature selection ARM approach yields feature that represent both normal and attack records. However, decision engine algorithms struggle to discern between normal and attack rows due to their comparable values.	The reliance on a small set of 100 rules for feature selection, potentially overlooking crucial attributes. The strict requirement of precisely 11 features may limit adaptability, and the lack of extensive validation across various datasets could compromise generalizability.
3	Intrusion Detection on AWS Cloud	2023	RM & MK	PCM + FCM-SMO + AE technique accuracy = 95.3%. on	The study analysed existing and new strategies across four assault categories

	through Hybrid Deep Learning Algorithm			CSE-CIC-IDS-2018 dataset.	using various criteria, resulting in 480 statistics in total. However, potential biases, dataset-specific overfitting, and limited generalizability to real-world circumstances are significant drawbacks, emphasising the importance of careful interpretation of the results.
4	An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique	2022	M.Mayuranathan, S.K. Saravanan, B. Muthusenthil, A. Samydurai	The proposed DKNN classifier outperformed current state-of-the-art LSTM-SGDM, LSTM-ADAM, CNN, CNN-LSTM, RC-NN, and DKNN classifiers in terms of accuracy, TPR, TNR, precision, recall, and F-measure.	The study demonstrates the effectiveness of the proposed EOS-IDS model using DARPA-IDS and CSE-CIC-IDS2018 datasets. However, limitations include potential dataset biases, reliance on specific pre-processing methods and algorithms, and the absence of real-world deployment validation.
5	Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning	2021	Ammar Aldallal, Faisal Alisa	Outperformed benchmarks by up to 5.74% using CICIDS2017 dataset. Achieved a maximum detection rate of 100% with 20 optimal features. Improvement ranged from 3.32% to 5.14% compared to previous works using different datasets.	The study showcased the efficacy of SVM kernel functions and feature selection for intrusion detection, albeit with potential constraints related to dataset-specific nuances and methodological choices. Comparisons with prior works may be influenced by varying experimental setups and metrics, underscoring the need for broader investigations with diverse datasets and methods to validate the findings.
6	Machine Learning Based Hybrid Intrusion Detection for Virtualized Infrastructures In Cloud Computing Environments	2021	Ayesha Sarosh	Better accuracy compared to earlier approaches	The hybrid intrusion model of K-means and SVM ensures accurate and quick detection, however it may have scaling limitations. Correlating application and network logs improves detection but may increase computational effort.
7	DDoS attack detection using machine learning techniques in cloud computing environments	2017	M. Zekri, S. E. Kafhali, N. Aboutabit, Y. Saadi	High detection accuracy (C4.5: 98.8%, Naïve Bayesian: 91.4%), Low false positives and false negatives, Efficient detection rate, Comparative analysis showing superiority of C4.5 algorithm for DDoS detection.	The experiment detects DDoS attacks using C4.5 classification with a promising detection rate of over 98%. However, limitations include the inability to simulate large cloud networks and reliance on a virtual environment instead. The use of parameterized python-scripts for generating normal network traffic may not fully replicate real-world scenarios.
8	Hybrid Deep Neural Architecture for Detection of DDoS Attacks in Cloud Computing	2021	Aanshi Bhardwaj, Veenu Mangat, Renu Vig	Achieved a detection rate of 95.74% and an accuracy of 98.25%.	The suggested technique detects DDoS with high accuracy and speed; however it is prone to dataset overfitting and lacks generalizability outside the CICIDS2017 dataset. Further validation across several datasets and real-world contexts is required.
9	Cyber Intrusion Detection Using Machine Learning Classification Techniques	2020	Hamed Alqahtani	Random Forest classifier consistently outperforms others in terms of accuracy, precision, recall, and F1-score, reflecting cyber-attack patterns effectively.	The study recommends Random Forest for intrusion detection because of its higher performance across metrics. However, relying on a single dataset restricts generalizability and may miss developing dangers. Integrating recency-based or contextual models may improve system efficacy.
10	Cloud-based Real-time Network Intrusion Detection Using Deep Learning	2018	Santhosh Parampottupada, Arghir-Nicolae Moldovann	H2O deep learning models using cross-validation achieved over 99.5% accuracy on the training dataset and over 83% accuracy on the test dataset for both binomial and multinomial classification.	The study compares binomial and multinomial models for intrusion detection. Binomial models excel in accuracy but lack generalization, while multinomial models perform decently but lack consistency across attack classes.
11	Designing an efficient security framework for detecting intrusions in virtual network of cloud computing	2019	Rajendra Patil, Harsha Dudeja, Chirag Modi	Detection of intrusions with high accuracy (more than 97% intrusive connections detected in real-time simulation), Low false positives (<0.5% false positive rate), Comparative analysis with existing approaches showing improved performance	The HLDNS framework shows promising results in detecting intrusions in both real-time simulation and offline validation. However, limitations include potential performance variations in dynamic network environments, increased computational costs with larger datasets, reliance on the quality of training data, and the need for further validation across diverse network architectures.
12	Detecting and Classifying Incoming Traffic in a Secure Cloud Computing Environment Using Machine Learning and Deep Learning	2022	Geetika Tiwari, Ruchi Jain	Results showed enhanced attack detection and increased classification accuracy up to 97%. Future work aims to explore advanced packet profiling and improve categorization performance through methods like data	Reliance on a single dataset, UNSW-NB15, potentially limiting real-world applicability. Evaluation metrics may oversimplify performance assessment. Lack of validation on diverse datasets or real cloud environments hampers generalizability. Additionally, practical deployment and resource constraints are not fully explored.

	System			mining and clustering.	
13	Machine Learning Based Intrusion Detection System in Cloud Environment	2022	Muhammad SalmanSaeed, Raman Saurabh, Sarang Balasaheb Bhasme, Alexey N.Nazarov	The study proposes a security framework for cloud intrusion detection, with a focus on DDoS attacks. It suggests machine learning, namely Random Forest, which achieves 99.99% accuracy. It focuses on real-time detection and the exploration of various machine learning algorithms to improve cloud security.	The paper mainly concentrates on DDoS attacks, neglecting other intrusion types. The evaluation could be more comprehensive with diverse datasets and attack scenarios.
14	Machine Learning Approach for Anomaly-Based Intrusion Detection Systems Using Isolation Forest Model and Support Vector Machine	2023	K. Shanthi, R. Maruthi	The research compared isolation forest and SVM for anomaly detection, with isolation forest showing slightly better performance. Anomaly-based machine learning offers fast, accurate detection, depending on proper feature selection.	It doesn't address the scalability of the proposed approach for larger datasets. Does not consider the impact of different anomaly types on detection performance
15	Intrusion Detection in Cloud Environments using Hybrid Deep Learning	2024	Kunal Rana	The experiments emphasised the relevance of algorithm selection and feature extraction strategies, with autoencoders demonstrating potential for enhancing classification accuracy with Bi-LSTM with auto encoder performing the best with 86% accuracy.	Limitations include dataset reliance, algorithm focus, metric selection, and interpretability issues. These concerns could be addressed by diversifying datasets, experimenting with different methods, and improving interpretability.

3 Research Methodology

In my research, I have used supervised and unsupervised learning. Supervised learning implies that the algorithm is privy to the expected outcome. In other words, through the input data, the mathematical model is trained for making accurate predictions. This type of learning is known as supervised learning and it utilizes data from a dataset, both the input and output. For example, the application of supervised learning can be employed to justify whether an email is genuine or not (0/1 situation).

The research design applied in this study aims at systematically achieving the major research goals of this study, which is to build and test the efficiency of machine learning and deep learning algorithms in intrusion detection on cloud computing systems. The basic preparation involves the installation of essential libraries and the loading of datasets using the Pandas library. The next step involves rigorous data cleaning procedures to handle the missing values and ensure that the cleaned data is of good quality. Plots and visualizations are then performed to explain the features of the dataset by applying tools from exploratory data analysis. Following the data preparation process there are several pre-processing techniques used for enhancing the usefulness of the data in the model, these include; label encoding, Min-Max scaling for normalization, and label binarization. To handle the problem of class imbalance, Synthetic Minority Over-Sampling Technique or SMOTE is applied. To guarantee a robust model evaluation, the dataset is then divided into training and testing sets at a ratio of 90:10. This account relies on the following sources: The process of feature selection and enhancement involves employing AutoEncoder neural networks which reduce the dimensionality a given dataset while preserving vital details. This step is necessary in order to optimise the performance and efficiency of the above model. Some of these are decision trees, logistic regression, and LSTMs and BiLSTMs which are different types of architectures in Machine learning and Deep learning. Standard model performance evaluation metrics such as confusion matrix and classification report are used. These metrics explain how accurately the models predict the cybersecurity outcomes that needs to be fixed for

creating robust security system resistant to the constantly evolving threats. (M. Mayuranathan, 2022).

My novelty in the research comprises using Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) with autoencoders to identify intrusions in cloud settings. This solution overcomes numerous issues inherent in standard intrusion detection systems by using the capabilities of recurrent neural networks (RNNs) and autoencoders.

Recurrent Neural Networks are special artificial neural networks that are used to process sequential data while keeping information from the previous steps. In contrast to feedforward neural networks, which work on inputs separately, RNNs contain connections that form directed loops, so they can incorporate temporal characteristics. The main characteristic of RNNs is the presence of memory or state of the previous inputs so they can model sequential data. This memory mechanism allows the RNNs to utilize information from previous time steps to make a prediction or decision on the present data. Another major drawback of most basic RNNs is the gradient vanish or explode problem which hinders the ability of the network to develop long-term dependencies during the training phase. To overcome this difficulty, other types of RNNs such as Long Short-Term Memory (LSTM) network have been proposed.

Autoencoders work such that they are able to come up with a simplified and more informative version of the input data. The technique utilized training of autoencoder models to learn first and second-order statistics features of the unlabeled network traffic data of both normal and anomalous traffic. These learnt properties, are feed into the LSTM and Bi-LSTM models to enhance the performance of intrusion detection.

The study is concerned with intrusion detection in cloud environment taking into consideration challenges typical for this environment. Specificity, constraints, and inherent volatility of cloud environments are crucial aspects to bear in mind when it comes to designing customized intrusion detection algorithms for the cloud.

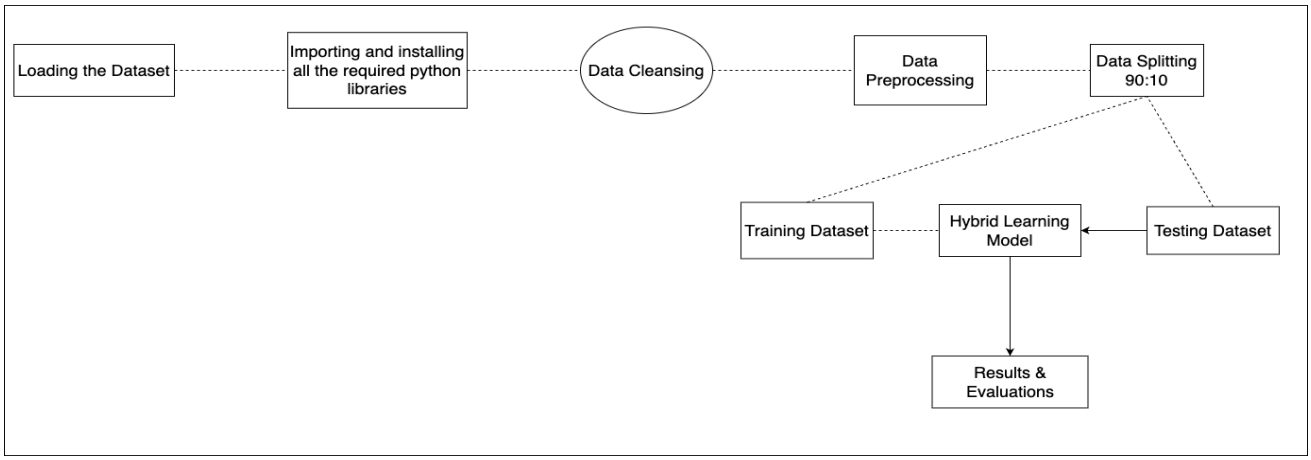


Figure 1: Process flow of proposed model

Step 1: Dataset acquisition

In this stage, all the libraries needed for data manipulation, analysis, and machine learning model implementation are imported and installed. NumPy, Seaborn, Matplotlib, NumPy, and Scikit-learn are among the frequently used libraries. The dataset is brought into the programming environment using the Pandas library. Pandas is well-suited for managing structured data, such as CSV files, because it offers robust capabilities for data analysis and modification.

Step 2: Data pre-processing and Visualization

This process removes null or missing values in the dataset hence is regarded as data cleaning process. To achieve this, missing data may be drop down to rows or columns, or values can be imputed employing techniques such as mean or median imputation. The process of gaining knowledge about the set of variables and their distributions is called exploratory data analysis or EDA. 'Loans and deposits' refers to money given and received by a company for a fixed period of time usually for business operations. When preparing the data to be used for training the model, certain pre-processing steps are taken. This involves performing label encoding to transform a categorical feature into numerical, normalizing the numerical features with the Minmax Scaler, and binarizing the labels if necessary.

Step 3: Data Splitting

In order to correct the imbalance in class, the Synthetic Minority Over-sampling Technique (SMOTE) oversamples the minority class. To reduce class imbalance and increase model correctness, this approach generates new samples. To assist in assessing the performance of the model, the dataset was split into test and training sets. It is common practice to divide the data into 90:10 training and test data sets in order to guarantee efficient training and validation..

Step 4: Model Training & Evaluation

I used autoencoder neural networks for feature selection and augmentation. This method improves the learning and efficiency of the model by reducing the dimensionality of the input characteristics while maintaining crucial information. Common assessment tools like the confusion matrix and classification report are used to evaluate the model's performance. These metrics give information on the model's recall, accuracy, precision, and F1-score, enabling a thorough evaluation of the model's performance.(Moldovann, 2018).

4 Algorithms

Algorithms are essential for creating and implementing intrusion detection systems (IDS) in AWS cloud settings, which improve network security. These algorithms cover a wide spectrum of methodologies, ranging from complex deep learning structures to conventional machine learning classifiers. Researchers want to improve the capacity to identify and successfully counteract cyber-attacks by utilising algorithms like Logistic Regression, Decision Tree Classifier, Long Short-Term Memory (LSTM), Bidirectional Long Short-Term

Memory (BiLSTM), and hybrid combinations thereof. These algorithms provide strong solutions for intrusion detection and prevention and have been carefully chosen and customised to match the dynamic and ever-evolving nature of security concerns in cloud systems. Researchers work to improve the overall security posture of AWS cloud infrastructures by applying algorithms methodically and protecting vital data and resources from a wide range of possible attacks (Zekri, 2017).

4.1 Logistic Regression

Logistic Regression is a fundamental algorithm employed primarily in binary classification tasks, where the objective is to predict the likelihood that an observation belongs to a specific class. It functions by creating a connection between one or more independent variables—often referred to as features—and the dependent binary variable, which represents the class labels (e.g., 0 or 1). The core function of a logistic regression is to estimate the probability of the binary result, which is commonly expressed as the likelihood of falling into the positive class (class 1, for example). The logistic function, sometimes referred to as the sigmoid function, is used to estimate this likelihood. A few benefits of using logistic regression are its ease of use, interpretability, and effectiveness when working with data that can be divided into linear segments. It works especially effectively in situations when a linear function may be used to describe the decision border between classes. Additionally, a deeper comprehension of the underlying relationships within the data is made possible by the insights that Logistic Regression offers into the influence of various features on the predicted probabilities. Because of these qualities, binary classification jobs in a variety of industries, such as marketing, finance, and healthcare, frequently employ logistic regression (Jain, 2022).

4.2 Decision Tree Classifier

The Decision Tree Classifier is an adaptable method that creates a hierarchical structure in feature space for classification and regression applications. Leaf nodes provide expected class labels, and each internal node indicates a judgement based on features. Its interpretability makes decision pathways understandable to stakeholders, which is important in industries like banking and healthcare. Nevertheless, decision trees can overfit, particularly when dealing with intricate datasets. Methods like as pruning cut off extraneous branches, simplifying and improving generalisation. Multiple trees are combined in ensemble approaches, such as Random Forests, to reduce overfitting and increase accuracy. Decision trees are widely used as machine learning basic models due to their interpretability and simplicity, even in the face of difficulties. They are an essential tool in data analysis and predictive modelling because they provide insightful information about patterns in data and can be applied to a wide range of areas and data types (Jain, 2022).

4.3 Long-Short Term Memory (LSTM)

Long Short-Term Memory, or LSTM, is a major development in recurrent neural network (RNN) architecture that was created expressly to solve the vanishing gradient issue that plagues conventional RNNs. Specialised memory cells with three gates—input, forget, and output—are the basis of LSTM operation. By controlling the input flow inside the network, these gates allow LSTM models to selectively eliminate irrelevant data while retaining critical information across lengthy periods. Because LSTMs can capture and retain long-term relationships, they are very useful for processing sequential data in applications like speech recognition, natural language processing, and time series prediction. Because LSTMs overcome the drawbacks of conventional RNNs, they perform better at recognising complicated sequences and capturing temporal trends. This makes them essential tools for a wide range of applications, including intrusion detection in AWS cloud settings. Because of their capacity to learn from sequential data, intrusion detection systems are more resilient overall by enabling precise identification of network abnormalities and possible security risks (M. Mayuranathan, 2022).

4.4 Bidirectional Long-Short Term Memory (Bi-LSTM)

Bidirectional Long Short-Term Memory, or BiLSTM, is an addition to the LSTM architecture that improves sequential data modelling and comprehension. BiLSTMs interpret input sequences concurrently using information from both past and future time steps, in contrast to typical LSTMs that solely take into account past data. BiLSTMs use backward and forward processing to handle input sequences, allowing them to extract context from both the items that come before and after them in the sequence. This two-way technique greatly enhances the network's comprehension and modelling of data relationships. BiLSTMs are widely used in jobs like natural language processing where contextual awareness is crucial. In applications like machine translation, named entity identification, and sentiment analysis, where obtaining the context from surrounding words is essential for precise interpretation and prediction, they perform exceptionally well. By using data from both past and future data points, BiLSTMs provide improved capabilities for comprehending network traffic patterns and spotting possible security risks in the context of intrusion detection in AWS cloud systems (M. Mayuranathan, 2022).

5 System Architecture

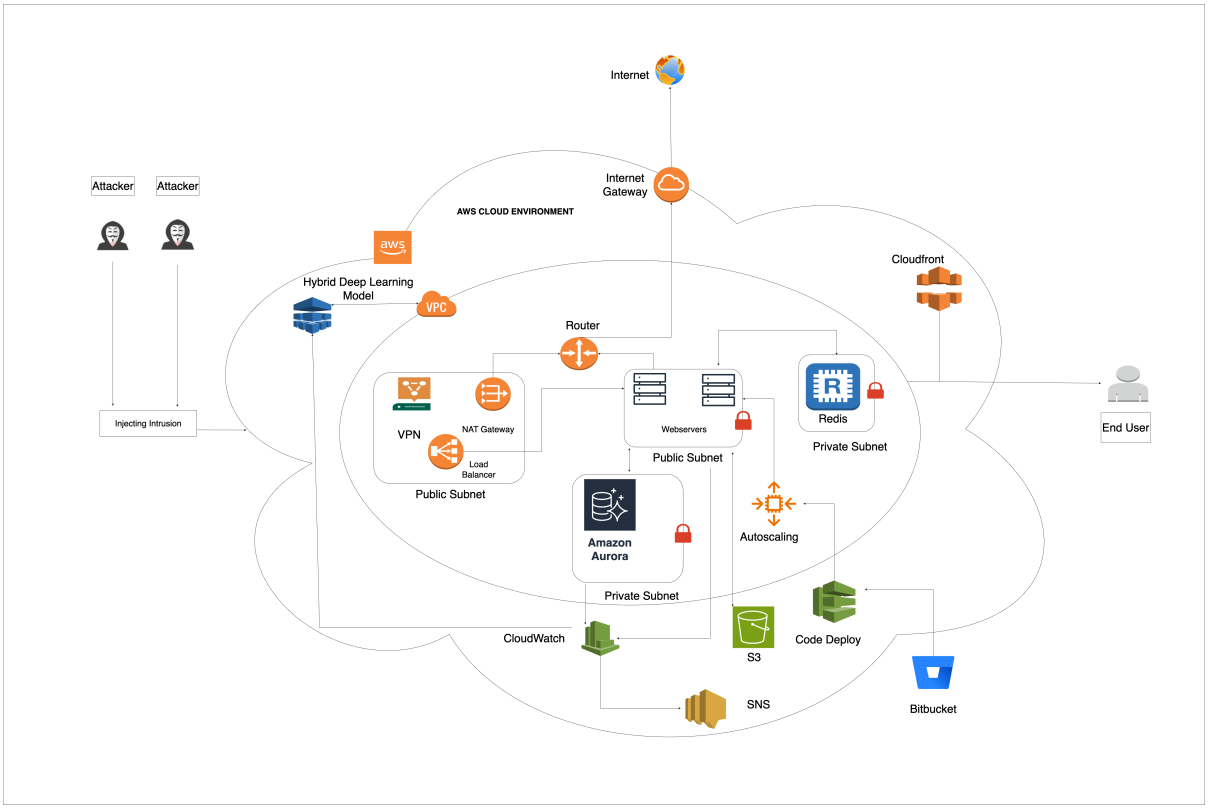


Figure: Architecture Diagram

The Virtual Private Cloud (VPC) framework, which offers a safe and separate environment within the cloud infrastructure, is the foundation of the recommended design. To provide high availability and fault tolerance, it is composed of four subnets that are divide over several availability zones. Two of these subnets are open to the public and are home to resources that may be accessed over the internet, including web servers, VPN servers, load balancers, and NAT gateways. These parts follow security best practices and allow for external communication.

The other two subnets are private and include sensitive resources such as Elastic Cache and the database. Placing these resources in private subnets increases security by limiting direct internet access, protecting critical data from unauthorized access.

A public route table that is linked to an internet gateway facilitates internet connectivity. Additionally, internet access for resources in the private subnet is provided via a NAT Gateway that is linked to the private route table.

Security Groups control incoming and outgoing network traffic for a variety of components, including the load balancer, Elastic Cache, VPN server, web server, and RDS (Relational Database Service). These security groups implement access controls by allowing or refusing communication according to predefined rules and protocols.

The autoscaling group handles web servers in the public subnet, scaling them dynamically based on demand to maximize resource utilization and provide constant performance during peak traffic periods.

AWS Code Deploy simplifies continuous integration and deployment, making it possible to update and publish web applications seamlessly. This ensures that new features or repairs may be implemented efficiently without affecting the application's availability.

The database server securely stores application data, while Elastic Cache is used for caching to improve speed by reducing database load and response times.

CloudFront, a content delivery network (CDN), is used for content caching to improve website performance by delivering material to users from edge locations that are closer to them, lowering latency and enhancing user experience.

CloudWatch monitors the overall health and performance of the system, providing information on resource utilization, application metrics, and system logs. Logs are kept in an S3 bucket for analysis and compliance, with auditing and troubleshooting capabilities available as needed.

The machine learning model is trained on historical network traffic data, which covers both normal and attack scenarios. Once trained and validated, the model is integrated into the architecture using Docker container to conduct real-time inference on incoming network traffic.

As fresh network packets are acquired by monitoring tools like CloudWatch Logs, they go through preliminary preprocessing steps with the help of an automated pipeline with pre-written scripts. These preparation stages are critical to ensure data consistency, quality, and readiness for further analysis. The automated pipeline performs activities such as handling missing value, removing duplicates, and data normalization, preparing it for the machine learning model's classification step.

The architecture is intended to be extremely scalable, allowing the machine learning model to rapidly analyze large amounts of network traffic in real time. Scalability guarantees that the system can withstand changes in incoming traffic volumes while maintaining performance and responsiveness. Additionally, batch processing methods and distributed computing frameworks can be used to improve the model's throughput and efficiency. By combining these methodologies, the system can identify network attacks quickly and accurately while minimizing latency, hence improving overall cybersecurity defenses in cloud settings.

6 Implementation

In my research, the acquisition and utilization of a complex UNSW-NB15 dataset which comprises of a hybrid mix of real-world network activity and simulated attack behaviors, the UNSW-NB15 dataset provides a thorough picture of the complexity of cloud infrastructure security. This dataset, which is a huge collection of network packets produced at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), offers a special chance to assess intrusion detection systems in settings similar to actual cloud installations. There are nine different kinds of attacks in this dataset: worms, reconnaissance, shellcode, DoS, backdoors, fuzzers, and exploits. In order to provide a total of 49 features with the class label, twelve algorithms are built, and the Argus and Bro-IDS tools are utilised. There are 175,341 records in the training set and 82,332 records in the testing set, which includes both normal

and other attack kinds. The significance of this dataset stage lies in its contribution to access intrusion detection in cloud environments, establishing it as not only important but also the initial step in the research process (Sarosh, 2021).

Dataset Loading and Cleaning:

The import of necessary libraries, such NumPy, Pandas, and Seaborn, which offer powerful data manipulation and visualisation capabilities, is the first step in the dataset loading process. The UNSW-NB15 dataset, which is kept in a CSV file format, is loaded using the Pandas package. After which I moved on to data cleaning which involves several essential stages in order to guarantee data integrity and get the information ready for analysis. Columns that are deemed unnecessary, such "service" and "id," are removed. In addition, the target column is updated to exclude instances of the 'Analysis' attack category in order to highlight more pertinent assault kinds. To find possible data inconsistencies, the presence of missing values is also evaluated. Finding patterns and trends in the dataset's features and attributes is the goal of visual analysis. First, to comprehend the numerical and category character of the data, the unique values in every column are looked at. The distribution and frequency of particular traits, like "attack categories" and "transaction protocols," are then visualised using box plots and bar charts as shown below in **Fig.2 and Fig.3** (Moustafa, 2015).

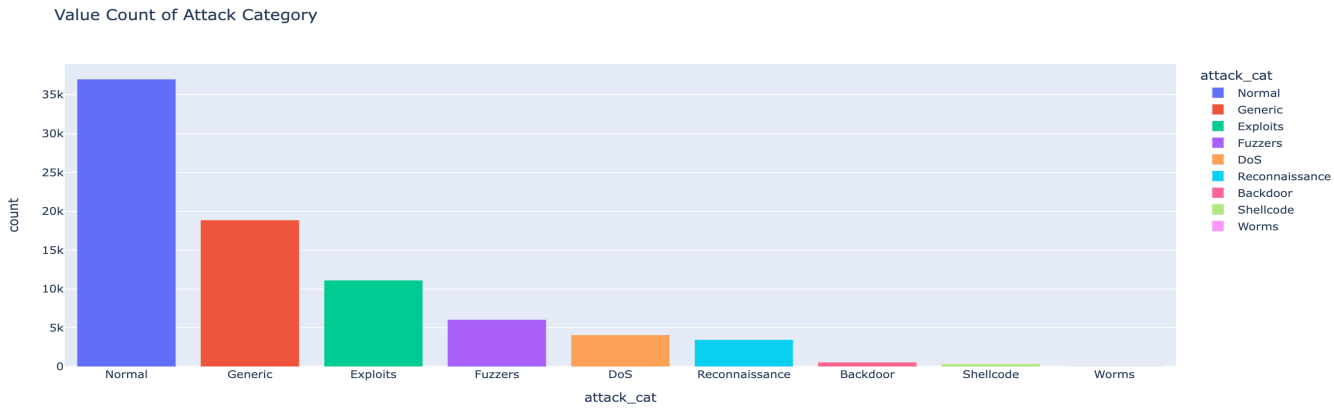


Figure 2: Value count of different attack categories.

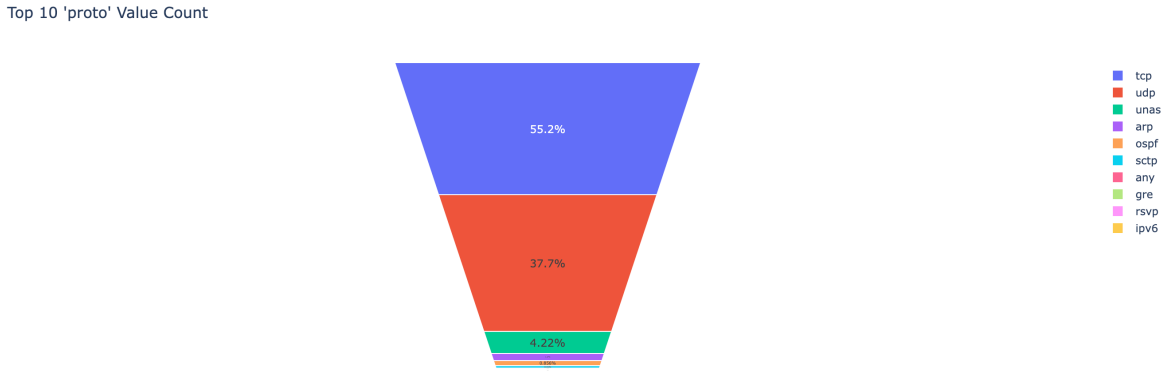


Figure 3: Value count of different protocols

Dataset Pre-processing:

After loading and cleaning the dataset, the second step involves data pre-processing.
First, label encoding is used to transform the feature set's categorical variables into a numeric representation. In order to investigate the linear relationship between characteristics, a correlation matrix is computed. It helps in locating strongly linked or redundant features, which may have an impact on model performance. The range of feature values between 0 and 1 is normalised for the feature set using Min-Max scaling. This helps in ensuring that every feature contributes equally to the training of the model and that features with bigger scales do not control the learning process. Synthetic Minority Over-sampling Technique (SMOTE) is used to balance the dataset. To address class imbalance difficulties, this strategy creates synthetic samples for the minority class (less common attack categories) so that the model learns from a more representative dataset as shown in **Fig.5**.

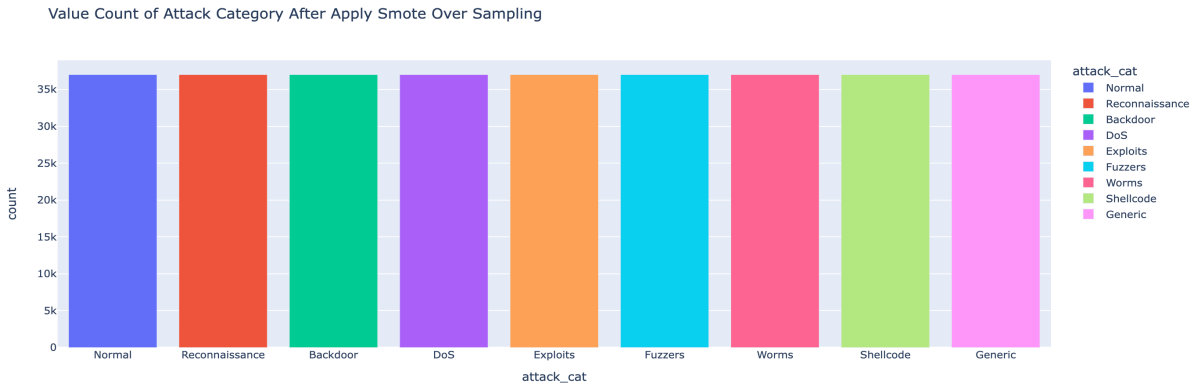


Figure 5: Value count of different attack categories after applying SMOTE

Feature Selection:

The selection of features is one of the most crucial steps in the machine learning and deep learning model training process. Feature selection is used in this study to make sure the model is trained with the best possible set of features—removing any that are considered unimportant. The method used for feature evaluation is correlation-based feature selection. Based on how well a feature correlates with the target variable, it evaluates a dataset. Weakly correlated features are deemed less predictively useful than those that are strongly correlated with the target variable. The correlation method groups features according to how closely they resemble the target variable, analysing the relationship between each feature and the intended result. It then chooses a subset of features to give the machine learning model the most pertinent characteristics (M. Mayuranathan, 2022).

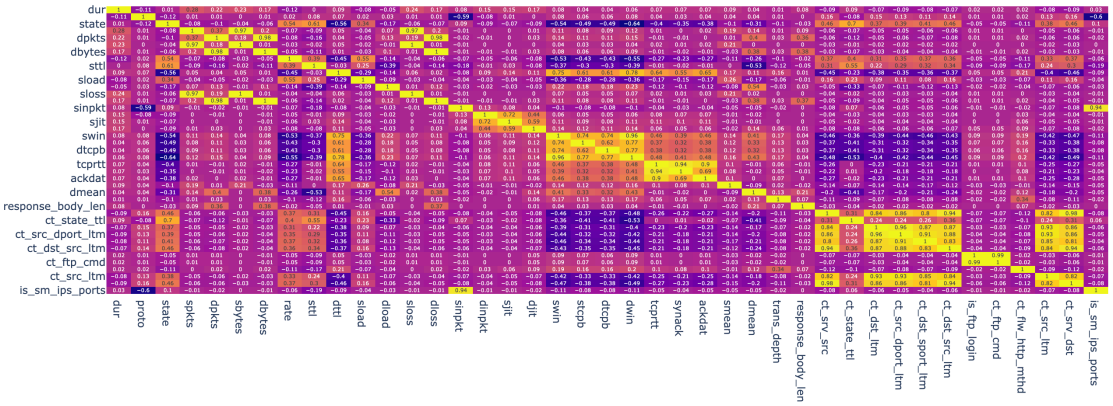


Figure 6: Statistics of correlation matrix

Models (Training & Testing):

The train-test split was the method used for data partitioning, which divided the labels and data. 10% was set aside for testing and 90% of the dataset was allocated for training. Building a hybrid learning model for intrusion detection in cloud environments was the aim of this project. The hybrid learning approach incorporates Long-Short-Term Memory (LSTM), Decision Tree, Bidirectional Long-Short-Term Memory (BiLSTM), and Logistic Regression classifiers. After developing the model, the split data and labels were fitted to the hybrid learning model that was trained on the training dataset. After the model was trained and saved, the testing dataset was used to assess the model's accuracy and F1 score.

During the last stage, the system was tested using the testing dataset. Ten percent of the original dataset was set aside for testing, and the other ninety percent was used for system training. The testing findings are presented below in the evaluation section including the specific results from each algorithm.

7 Evaluation

The results of applying and assessing machine learning and deep learning models for Intrusion detection in cloud environments are shown in this part, along with metrics like accuracy, F1-score, and confusion matrix. Based on the data, Logistic Regression was able to attain an accuracy rate of 62%. Decision Tree Classifier showed the least accuracy, with F1 score of 53%. LSTM when trained and tested without autoencoder was able to achieve an accuracy of 82%. BiLSTM without the use of autoencoder reaches accuracy rate of 83%. With an F1 score of 80% and accuracy of 81%, there was not much difference of using autoencoders with the LSTM model. With an 86% accuracy rate and an F1 score of 86% for intrusion detection, BiLSTM with autoencoders demonstrated remarkable proficiency in detection cyber-attacks. The outcomes of every algorithm include a Confusion Matrix along with related measures such as accuracy, recall, precision, f1-score, weighted average, and macro-average.

Accuracy - Accuracy measures the proportion of correctly classified instances out of the total instances. It is calculated as the ratio of the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{TruePositives + True\ Negatives}{Total\ Predictions}$$

Precision - Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall (Sensitivity) - Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

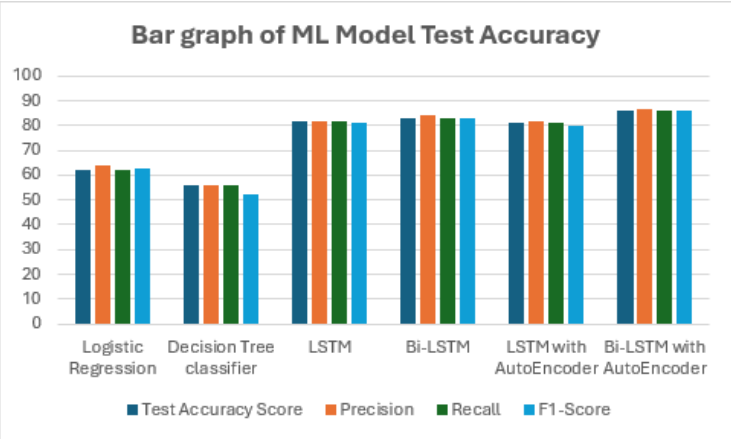
F1-score – The F1-score is the harmonic mean of precision and recall, providing a balance between these two metrics.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Confusion Matrix - A confusion matrix provides a tabular representation of the actual vs. predicted classes by the model. It includes four values: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
True Positive (TP): The percentage of values correctly identified as true by the model, considering both the actual truth and the predicted outcomes.
True Negative (TN): The true negative represents the percentage of values that are genuinely negative, and the model correctly predicts a negative outcome as well.
False Positive (FP): The False Positive is the count of truly negative values for which the model predicted a positive outcome.
False Negative (FN): The false negative is the percentage of truly negative values that the model erroneously predicted to be true (Azawii, 2019).

	Actual True/False	
Predicted Positive/Negative	True Positive	False Positive
	False Negative	True Negative

ML & DL MODEL	Test Accuracy Score	Precision	Recall	F1-sccore
Logistic Regression	62	64	62	63
Decision Tree Classifier	56	56	56	52
LSTM	82	82	82	81
Bi-LSTM	83	84	83	83
LSTM with AutoEconder	81	82	81	80
Bi-LSTM with AutoEncoder	86	87	86	86



7.1 Experiment 1: Evaluating Logistic Regression Classifier

The experiment attempts to assess the Logistic Regression classifier's performance in terms of precision, recall, and F1 score. The confusion matrix visualises the classifier's capacity to categorise cases. Rows indicate expected labels, with "Backdoor" denoted by zero, “DoS” denoted by one, “Exploits” denoted by two, “Fuzzers” denoted by three, “Generic” denoted by four, “Normal” denoted by five, “Reconnaissance” denoted by six, “Shellcode” denoted by seven and "Worms" by eight. Columns correspond to the actual labels and rows correspond to predicted labels. True Positive (TP) and True Negative (TN) is represented by counts of 2643, 1732,1784, 2162, 3344, 2291, 1751, 2136 and 2940 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 428, 291, etc for backdoor and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 269, 88, etc for backdoor and so on for respective labels.
The bar graph above shows significant metrics from the classification report. Precision, recall, and F1 scores are provided for each class. For backdoor attacks (class 0), precision is 63%, recall is 71%, and F1 score is 67. For DoS attacks (class 1), precision is 54%, recall is 47%, and F1 score is 51. For Exploits (class 2), precision is 66%, recall is 49% and F1 score is 56. For Fuzzers (class 3), precision is 60%, recall is 58% and F1 score is 59. For Generic (class 4), precision is 83%, recall is 88% and F1 score is 85. For normal (class 5), precision is 93%, recall is 63% and F1 score is 75. For reconnaissance (class 6), precision is 39%, recall is 47% and F1 score is 42. For shellcode (class 7), precision is 48%, recall is 59% and F1 score is 53. For worms (class 8), precision is 71%, recall is 79% and F1 score is 75.
These visualisations provide a succinct overview of the Logistic Regression Classifier's ability to discriminate between different types of attacks , providing useful insights into its classification performance.

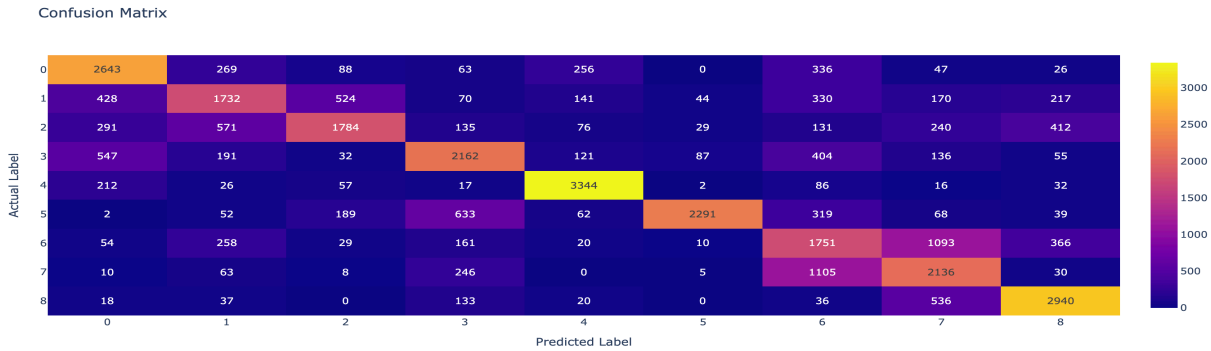


Figure 7: Confusion Matrix Logistic Regression

. . .

7.2 Experiment 2: Evaluating Decision Tree Classifier

The goal of this experiment is to determine the usefulness of the Decision Tree classifier by assessing its precision, recall, and F1 score. The confusion matrix for the Decision Tree classifier provides crucial performance parameters for identifying different types if attacks. Columns correspond to the actual labels and rows correspond to predicted labels. True Positive (TP) and True Negative (TN) is represented by counts of 2927, 1408, 524, 2439, 3594, 1817, 0, 3003 and 2959 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 456, 284, 164, etc for DoS and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 524, 336, 476, etc for DoS and so on for respective labels.

The classification report also displays the precision, recall, and F1 score for each class. For backdoor (class 0), precision is 62%, recall is 79% and F1 score is 69. For DoS (class 1), precision is 57%, recall is 39% and F1 score is 46. For exploits (class 2), precision is 47%, recall is 14% and F1 score is 22. For fuzzers (class 3), precision is 40%, recall is 65% and F1 score is 50. For generic (class 4), precision is 100%, recall is 95% and F1 score is 97. For normal (class 5), precision is 95%, recall is 50% and F1 score is 65. For reconnaissance (class 6), precision is 0%, recall is 0% and F1 score is 0. For shellcode (class 7), precision is 32%, recall is 83% and F1 score is 47. For worms (class 8), precision is 72%, recall is 80% and F1 score is 76%.

These data highlight the Decision Tree classifier's weak performance and ability in distinguishing between different types of attacks.

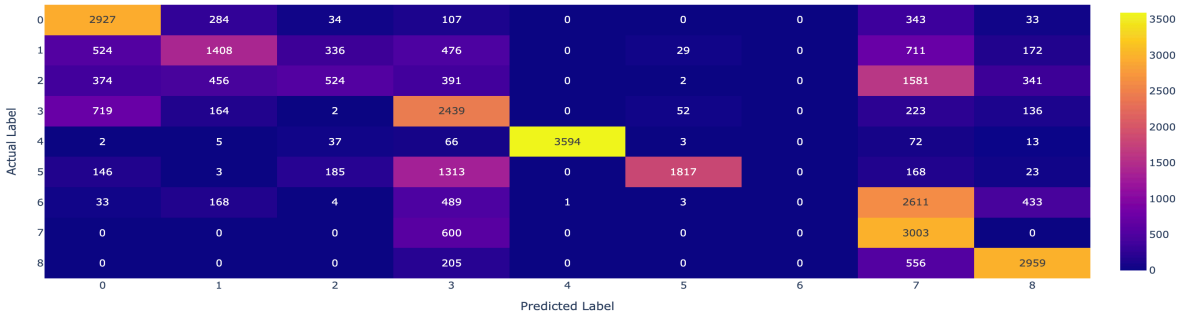


Figure 8: Confusion Matrix Decision Tree Classifier

. . .

7.3 Experiment 3: Evaluating Long-Short Term Memory(LSTM) without Auto Encoders

The goal of this experiment is to determine the long-short term memory (LSTM) algorithm's effectiveness by evaluating its performance parameters such as precision, recall, and F1 score. The LSTM's confusion matrix visually illustrates its ability to discriminate between different attacks. Columns show the actual labels (0 for Backdoor, 1 for DoS, 2 for Exploits, 3 for Fuzzers, 4 for Generic, 5 for Normal, 6 for Reconnaissance, 7 for Shellcode and 8 for Worms), whereas rows represent anticipated labels. Columns correspond to the actual labels and rows correspond to predicted labels. True Positive (TP) and True Negative (TN) is represented by counts of 3394, 2404, 2014, 2630, 3481, 3169, 3027, 3369 and 3652 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 10, 57, 5, etc for exploits and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 875, 120, 0, etc for exploits and so on for respective labels. The classification report refines these parameters. For Backdoor, precision is 70%, emphasising accurate positive predictions. The model's 93% recall indicates accuracy in detecting valid cases, resulting in an 80 F1 score. For DoS, the model had a 60% precision, indicating exact affirmative predictions, while a 66% recall captures a significant amount along with F1 score which is 63. For Exploits, precision is 75%, recall is 54% and F1 score is 63. For Fuzzers, a precision of 80%, recall 71% along with F1 score of 75. For Generic, precision is 100%, recall is 96% and F1 score is 98. For Normal, precision is 92%, recall is 86% and F1 score is 89. For Reconnaissance, precision is 85%, recall is 82% and F1 score is 83. For Shellcode, precision is 83%, recall is 90% and F1 score is 86. And lastly, for Worms, precision is 93%, recall is 97% and F1 score is 95. These data highlight the Long-Short Term Memory algorithm's robust performance and ability in distinguishing between different types of attacks.

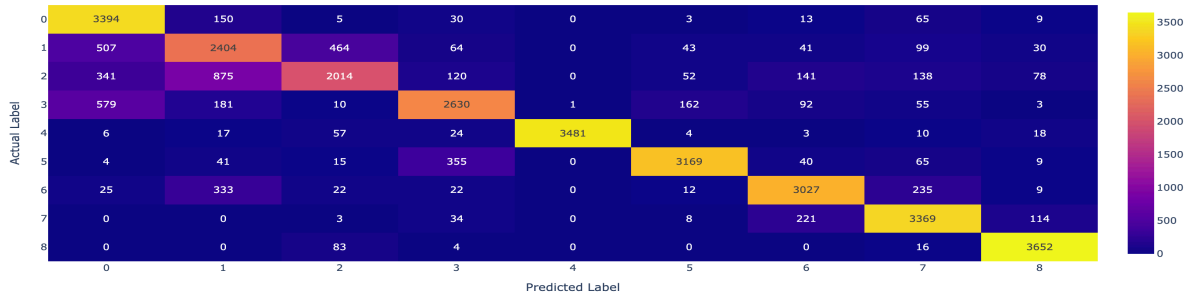


Figure 9: Confusion Matrix Long-Short Term Memory (LSTM)

. . .

7.4 Experiment 4: Evaluating Bidirectional Long-Short Term Memory (Bi-LSTM) without Auto Encoders

The purpose of this experiment is to assess the Bidirectional Long-Short Term Memory (Bi-LSTM) algorithm's usefulness by measuring performance parameters such as precision, recall, and F1 score. The confusion matrix visually represents the Bidirectional Long-Short Term Memory (Bi-LSTM) algorithm's performance in discriminating between different types of attacks. Columns represent actual labels, whereas rows represent anticipated labels. Columns correspond to the actual labels and rows correspond to predicted labels. True

Positive (TP) and True Negative (TN) is represented by counts of 3510, 2443, 2000, 2518, 3492, 3383, 3095, 3478 and 3719 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 17, 92, 62, etc for fuzzers and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 12, 3, 302, etc for fuzzers and so on for respective labels. In the classification report, precision for Backdoor attacks is 69%, demonstrating moderate accuracy in properly identifying them, while recall is 96%, suggesting the model's success in catching a large number of actual legitimate backdoor attacks. The F1 score for backdoor attacks is 80%, indicating a balanced performance. For DoS, the precision is 60%, recall is 67% and F1 score is 63. For Exploits, precision is 79%, recall is 53% and F1 score is 63%. For Fuzzers, precision is 87%, recall is 68% and F1 score is 76. For Generic, precision is 100%, recall is 96% and F2 score is 98. For Normal, precision is 89%, recall is 91% and F1 score is 90. For Reconnaissance, precision is 85%, recall is 84% and F1 score is 85. For Shellcode, precision is 88%, recall is 93% and F1 score is 90. For Worms, precision is 97%, recall is 99% and F1 score is 98, indicating a balanced approach to identifying different types of attacks.

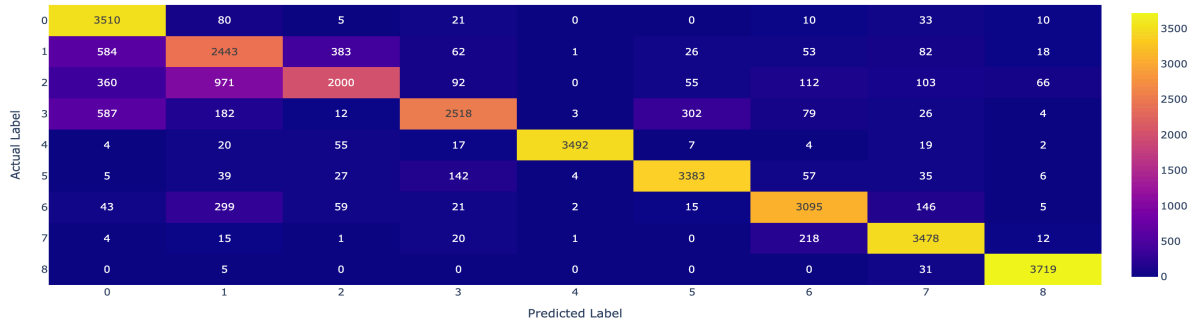


Figure 10: Confusion Matrix Bidirectional Long-Short Term Memory (Bi-LSTM)

. . .

7.5 Experiment 5: Evaluating Long-Short Term Memory with Auto Encoder

The goal of this experiment is to determine the effectiveness of the Long-Short Term Memory (LSTM) algorithm by evaluating its performance parameters such as precision, recall, and F1 score. The Long-Short Term Memory (LSTM) algorithm's confusion matrix visualises its ability to discriminate between different types of attacks. Columns reflect actual labels (0 for Backdoor, 1 for DoS, 2 for Exploits, 3 for Fuzzers, 4 for Generic, 5 for Normal, 6 for Reconnaissance, 7 for Shellcode and 8 for Worms), whereas rows indicate anticipated labels. Columns correspond to the actual labels and rows correspond to predicted labels. True Positive (TP) and True Negative (TN) is represented by counts of 3653, 2572, 1438, 2540, 3401, 3149, 2934, 3570 and 3602 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 24, 6, 0, etc for generic and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 54, 31, 4, etc for generic and so on for respective labels. The classification report improves upon these indicators. The precision for backdoor attacks is 66%. A 97% recall illustrates the model's ability to recognise authentic cases, resulting in a 79 F1 score. DoS attacks have a 56% precision, recall is 68% and F1 score 62. For Exploits, precision is 79%, recall is 38% and F1 score is 52. For Fuzzers, precision is 81%, recall is 69% and F1 score is 75. For generic, precision is 99%, recall is

94% and F1 score is 97. For Normal, precision is 91%, recall is 87% and F1 score is 89%. For Reconnaissance, precision is 82%, recall is 80% and F1 score is 81. For Shellcode, precision is 86%, recall is 95% and F1 score is 90. Lastly, for Worms, precision is 99%, recall is 98% and F1 score is 99, demonstrating the model's balanced precision and recall in most of the classes.

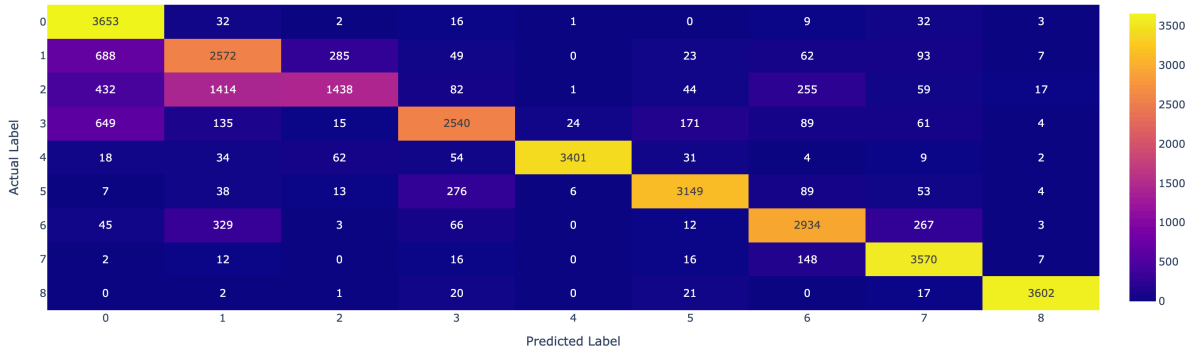


Figure 11: Confusion Matrix LSTM with AutoEncoder

. . .

7.6 Experiment 6: Evaluating Bidirectional Long-Short Term Memory with Auto Encoders

The purpose of this experiment is to assess the Bidirectional Long-Short Term Memory (bi-LSTM) algorithm with AutoEncoder by measuring performance parameters such as precision, recall, and F1 score. The confusion matrix visually represents the Bi-LSTM algorithm's performance in discriminating between different types of attacks. Columns represent actual labels, whereas rows represent anticipated labels. Columns correspond to the actual labels and rows correspond to predicted labels. True Positive (TP) and True Negative (TN) is represented by counts of 3656, 2959, 2197, 2711, 3469, 3309, 3082, 3626 and 3655 for the respective labels from backdoor till worms. False Positive (FP) is represented by the values present in the same column by counts of 7, 11, 143, etc for normal and so on for the respective labels. False Negative (FN) is represented by the values present in the same rows by counts of 145, 12, 65, etc for normal and so on for respective labels. In the classification report, precision for Backdoor is 71%, demonstrating accuracy in properly identifying them, while recall is 98%, suggesting the model's success in catching a large number of actual backdoor attacks. The F1 score for backdoor attacks is 82%, indicating a balanced performance. The algorithm accurately identified DoS attacks with a precision of 66% and recall of 78%. The F1 score for DoS attacks is 72%, indicating a balanced approach to identifying DoS attacks. For Exploits, precision is 84%, recall is 59% and F1 score is 69. The algorithm's precision for Fuzzer attacks is 91%, recall is 74% and F1 score is 81. For Generic attacks, precision is 99%, recall is 96% and F1 score is 98%. For normal connections, precision is 93%, recall is 91% and F1 score is 92%. For Reconnaissance, precision is 90%, recall is 84% and F1 score is 87%. For Shellcode attacks, precision is 92%, recall is 96% and F1 score is 94%. For worm attacks, precision is 98%, recall is 100% and F1 score is 99%, illustrating a balanced performance in classifying different types of attacks.

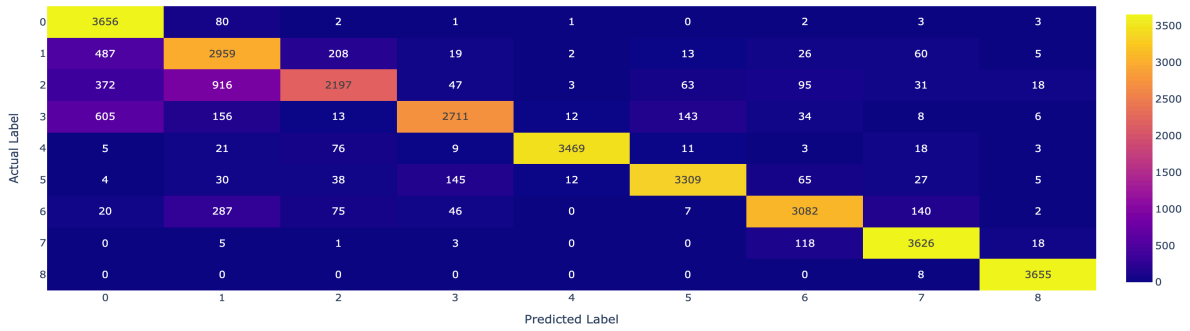


Figure 12: Confusion Matrix Bi-LSTM with AutoEncoder

. . .

7.7 Discussion

The experiments in this study sought to assess the performance of various machine learning algorithms, including Logistic Regression, Decision Tree Classifier, Long-Short Term Memory (LSTM), and Bidirectional Long-Short Term Memory (Bi-LSTM), in classifying different types of attacks in network intrusion detection. Each trial revealed the individual algorithm's strengths and flaws.

In Experiment 1, the Logistic Regression classifier performed moderately across the assault classifications. Precision, recall, and F1 scores varied between classes, with some outperforming others. For example, the classifier performed well for "Generic" and "Normal" classes, but poorly for "Reconnaissance" and "Shellcode." The confusion matrix revealed information on the classifier's ability to categorise distinct attack types, and the classification report summarised the precision, recall, and F1 scores for each class. Overall, the Logistic Regression classifier produced encouraging results; however, there is potential for improvement, particularly in classes with lower performance metrics.

In Experiment 2, the performance of the Decision Tree classifier varied among attack classes. Some courses obtained excellent precision, recall, and F1 scores, while others had lower performance measures. For example, the classifier performed well for "Generic" and "Normal" classes, but poorly for "Exploits" and "Reconnaissance." The confusion matrix revealed information on the classifier's ability to categorise distinct attack types, and the classification report summarised the precision, recall, and F1 scores for each class. Overall, the Decision Tree classifier performed inconsistently, showing the need for additional optimisation, and refining to improve its effectiveness in network intrusion detection.

In Experiment 3, the Long-Short Term Memory (LSTM) algorithm performed well across multiple attack types. Precision, recall, and F1 scores varied between classes, with some outperforming others. For example, the LSTM scored great precision and recall for "Generic" and "Normal" classes but performed poorly for "DoS" and "Reconnaissance." Overall, the LSTM algorithm without autoencoders produced encouraging results, pointing to its potential for network intrusion detection.

In Experiment 4, similar to the LSTM technique, the Bidirectional Long-Short Term Memory (Bi-LSTM) approach without autoencoders performed well across multiple attack classes. Precision, recall, and F1 scores varied between classes, with some outperforming others. For example, the Bi-LSTM performed well in "Generic" and "Normal" classes, but poorly in

"Exploits" and "Reconnaissance." Overall, the Bi-LSTM algorithm without autoencoders produced promising results, demonstrating its usefulness in network intrusion detection.

In Experiment 5, adding autoencoders to the LSTM algorithm enhanced its performance metrics across a variety of attack types. Precision, recall, and F1 scores improved significantly compared to the LSTM without autoencoders. Classes such as "DoS" and "Exploits" showed significant increases in precision, recall, and F1 scores, demonstrating that the LSTM with autoencoders can distinguish between different attack types. Overall, the LSTM with autoencoders outperformed the unencoded LSTM, demonstrating autoencoders' usefulness for feature extraction in network intrusion detection.

In Experiment 6, incorporating autoencoders into the Bidirectional Long-Short Term Memory (Bi-LSTM) method improves performance metrics across a variety of attack classes. Precision, recall, and F1 scores improved significantly compared to the Bi-LSTM without autoencoders. Classes such as "Exploits" and "Reconnaissance" showed significant increases in precision, recall, and F1 scores, demonstrating that the Bi-LSTM with autoencoders can distinguish between different attack types. Overall, the Bi-LSTM with autoencoders performed better, highlighting the efficacy of autoencoders in feature extraction for network intrusion detection.

8 Conclusion and Future Work

In this study, I sought to evaluate the performance implications of employing machine learning and deep learning algorithms for intrusion detection in cloud environments, and how do they contribute to improving the overall security posture of cloud infrastructures?

Throughout my research, I have successfully assessed a variety of machine learning methods, including Logistic Regression, Decision Tree, Long-Short Term Memory (LSTM), and Bidirectional Long-Short Term Memory (Bi-LSTM), for intrusion detection in cloud environments. I evaluated each algorithm's performance measures across distinct attack classes using a series of trials, including precision, recall, and F1 scores. The findings provide important insights into the effectiveness of these machine learning and deep learning methods for detecting intrusions in cloud environments. While some algorithms performed well in detecting specific attack types, others showed limitations in their usefulness. For example, algorithms like as LSTM and Bi-LSTM have shown promising results in differentiating between different attack classes, especially when combined with feature extraction approaches such as autoencoders. However, the research identifies several limitations and obstacles in using hybrid deep learning for intrusion detection in cloud systems. These include challenges with dataset imbalance, algorithm complexity, and the necessity for ongoing monitoring and adaption to changing threats. Despite these limitations, this study adds to the body of knowledge in cloud security by exploring the performance implications of hybrid deep learning methods for intrusion detection. Understanding the benefits and disadvantages of various algorithms enables organisations to make informed decisions when designing and implementing intrusion detection systems in cloud settings.

When evaluating future prospects, there are various options for improving intrusion detection capabilities in cloud systems. One possible approach is to use ensemble approaches such as Random Forests and Gradient Boosting to improve classification accuracy and robustness. Additionally, building intrusion detection systems that dynamically react to incoming threats using adaptive learning algorithms has the potential to dramatically improve detection

capabilities. Advanced deep learning architectures, such as Convolutional Neural Networks (CNNs) and Transformer models, may provide additional advances in pattern recognition. Furthermore, combining Explainable AI (XAI) approaches may improve the interpretability and transparency of intrusion detection systems. Real-world deployment and evaluation via field trials are required to prove the scalability and practical applicability of these solutions in live cloud environments.

In conclusion, while the research sheds light on the performance implications of using machine learning methods for intrusion detection in cloud systems, there is still more to be discovered and improved. Using the conclusions of this study, organisations can increase their security posture and better defend against emerging cyber threats in cloud infrastructures.

References

Azawii, Ali & Al-Janabi, Sufyan & Al-Khateeb, Belal. (2019). Survey on Intrusion Detection Systems based on Deep Learning. Periodicals of Engineering and Natural Sciences (PEN). 7. 1074-1095. 10.21533/pen.v7i3.635.

Moustafa, Nour & Slay, Jill. (2015). The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems. 25-31. 10.1109/BADGERS.2015.014.

Balajee, R. & Kannan, M.K.Jayanthi. (2023). Intrusion Detection on AWS Cloud through Hybrid Deep Learning Algorithm. 12. 1423.

M. Mayuranathan, S.K. Saravanan, B. Muthusenthil, A. Samydurai. An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. Advances in Engineering Software, Volume 173, 2022, 103236, ISSN 0965-9978.

Aldallal, Ammar & Alisa, Faisal. (2021). Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning. Symmetry. 13. 1-26. 10.3390/sym13122306.

Sarosh, Ayesha. (2021). Machine Learning Based Hybrid Intrusion Detection ForVirtualized Infrastructures In Cloud Computing Environments. Journal of Physics: Conference Series. 2089. 012072. 10.1088/1742-6596/2089/1/012072.

Zekri, Marwane & El Kafhali, Said & Aboutabit, Noureddine & Saadi, Youssef. (2017). DDoS attack detection using machine learning techniques in cloud computing environments. 1-7. 10.1109/CloudTech.2017.8284731.

Bhardwaj, A., Mangat, V., Vig, R. (2021). Hybrid Deep Neural Architecture for Detection of DDoS Attacks in Cloud Computing. In: Paprzycki, M., Thampi, S.M., Mitra, S., Trajkovic, L., El-Alfy, ES.M. (eds) Intelligent Systems, Technologies and Applications. Advances in Intelligent Systems and Computing, vol 1353.

Alqahtani, Hamed & Sarker, Iqbal & Kalim, Asra & Hossain, Syed & Ikhlaq, Sheikh & Hossain, Sohrab. (2020). Cyber Intrusion Detection Using Machine Learning Classification Techniques. 10.1007/978-981-15-6648-6_10.

S. Parampottupadam and A. -N. Moldovann, "Cloud-based Real-time Network Intrusion Detection Using Deep Learning," 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Glasgow, UK, 2018, pp. 1-8, doi: 10.1109/CyberSecPODS.2018.8560674. keywords: {Network security;intrusion detection;deep learning;cloud computing;NSL-KDD},

Rajendra Patil, Harsha Dudeja, Chirag Modi. Designing an efficient security framework for detecting intrusions in virtual network of cloud computing, Computers & Security, Volume 85, 2019, Pages 402-422, ISSN 0167-4048.

G. Tiwari and R. Jain, "Detecting and Classifying Incoming Traffic in a Secure Cloud Computing Environment Using Machine Learning and Deep Learning System," 2022 IEEE 7th International Conference on Smart Cloud (SmartCloud), Shanghai, China, 2022, pp. 16-21, doi: 10.1109/SmartCloud55982.2022.00010. keywords: {Deep learning;Training;Cloud computing;Machine learning algorithms;Firewalls (computing);Cloud computing security;Classification algorithms;Cloud computing;Intrusion Detection System;Machine Learning;Deep Learning;UNSW-NB-15.},

K. Shanthi and R. Maruthi, "Machine Learning Approach for Anomaly-Based Intrusion Detection Systems Using Isolation Forest Model and Support Vector Machine," 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2023, pp. 136-139, doi: 10.1109/ICIRCA57980.2023.10220620. keywords: {Support vector machines;Analytical models;Computational modeling;Memory management;Intrusion detection;Machine learning;Forestry;Intrusions detection;Anomaly-based;Machine Learning;Isolation Forest Model;Support Vector Machine},