National College of Ireland

Optimizing Network Security: Performance Analysis of Neural Network Models for Intrusion Detection

MSc Research Project
MSc in Cybersecurity

Ranjith Kumar Saravanan
Student ID: 22209751

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ranjith Kumar Saravanan		
Student ID:	22209751		
Programme:	MSc in Cybersecurity	Year:	2023-2024
Module:	Practicum		
Supervisor:	Khadija Hafeez		
Submission Due Date:	19.08.2024		
Project Title:	Optimizing Network Security: Performance Analysis of Neural Network Models for Intrusion Detection		
Word Count:	6005	Page Count: 19	

Signature: Ranjith Kumar Saravanan ………………………………………………………………………………………………………

Date: 19.08.2024 ………………………………………………………………………………………………………

Optimizing Network Security: Performance Analysis of Neural Network Models for Intrusion Detection

Ranjith Kumar Saravanan

22209751

Abstract

Since cybersecurity threats became more intelligent, defending against them required high-powered mechanisms to maintain network integrity. As such, the report devised potent intrusion detection models using the NSL-KDD dataset. The dataset included various vital threats and hazards, such as the back, neptune, portsweep, and smurf alongside normal traffic. The effectiveness of the Artificial Neural Networks and Long Short-Term Memory LSTM models was tested on these samples. The results demonstrated a validation accuracy of 66.70% for the ANN and 99.10% for the LSTM, offering a novel neural network approach to these major types of malicious shellcode, rather than depending on previous signature-based methods. The developed approach enabled outstanding accuracy and far fewer false positives than before, providing an improvement in cybersecurity.

1 Introduction

The need for more advanced and powerful intrusion detection systems is prompted by the continuous advancement of cybersecurity threats and the growth of their complexity and frequency. Although traditional IDSs based on signatures are the cornerstone of security measures, they fail to protect networks from new or complex attacks. Therefore, while optimizing the existing methods, it is also pertinent to pay closer attention to more dynamic approaches that will allow enhanced network security. In this context, the recent developments in machine learning have opened many opportunities for the development of efficient IDS. The use of neural networks, particularly Artificial Neural Networks, and LSTM, is one of the most promising developments to emerge. The high precision of these models in detecting and classifying the network attack has been witnessed in the study by Ali et al. (2024) used ANN in combination with the ABC approach was found to have the detection rate of up to 99%, which was one of the highest performance rates. The usage of neural networks, therefore, is expected to ensure the more precise distinction of the attacks and normal traffic, providing the opportunity to enhance the network's security. However, the use of an additional optimization algorithm, ABC, often brings an additional level of complexity, which may not always be acceptable. Therefore, the core objective of this research is the development of machine learning-oriented approaches to optimize the network security performance based on the application of purely satellite models and the comparison of their performance with an LSTM model that has already demonstrated high efficiency in sequence learning tasks.

Research Question

- *Does an ANN -based IDS or LSTM-based IDS perform more effectively in terms of accuracy, precision, and recall when applied to the NSL-KDD dataset?*

Objectives

1. To develop and evaluate an ANN-based IDS using the NSL-KDD dataset, assessing its accuracy and effectiveness in detecting various cyber threats.

2. To implement and test an LSTM-based IDS on the same dataset and compare its performance with that of the ANN model.

3. To analyze and understand the trade-offs between using standalone ANN models and LSTM models in terms of detection accuracy and computational efficiency.

By focusing on these objectives, this research aims to contribute to the field of network security by providing insights into the effectiveness of different machine learning models and potentially offering more streamlined solutions for intrusion detection.

2 Related Works

In cyber security, Intrusion Detection Systems are one of the most important facets in terms of cyber security. From 2020 to 2024, prominent research studies have enriched IDS through several ML/DL mechanisms, including Recursive Feature Elimination, Deep Neural Networks, Convolutional Neural Networks. These works have paved the road towards higher detection accuracy, fewer false-positives, and improved performance across different environments including IoT and cloud networks. In the future, the most profound direction will relate to adaptations to novel threats, improved practical significance, and successful combinations with AI approaches.

Feature Selection and Classification Techniques

Recursive Feature Elimination and Classification Models: Bilal Mohammeda and Ekhlas K. Gbashi (2021) used Recursive Feature Elimination (RFE) for feature selection and used Deep Neural Network (DNN) and Recurrent Neural Network (RNN) for the classification and the accuracy was found to be 94%. Feature Selection and Optimizers: Unsupervised anomaly detection was done using Isolation Forest or One Class Support Vector Machine (OCSVM) with active learning by Kavitha S et al. (2021) yielding higher accuracy than others. In recent work, Abdulaziz Fatani et al. (2021) employed the IDS accuracy improvement using CNNs and utilized a new feature selection technique based on TSO and DE operators. Lightweight anomaly detection has been introduced by Azam Davahli et al. (2023) which employed Support Vector Machine (SVM) and a hybrid of genetic algorithm and grey wolf optimizer for feature selection and computational optimization. A novel filter-based ensemble feature selection (FEFS) method along with deep learning model (DLM) based on recurrent neural network (RNN) optimizer with Tasmanian devil optimization (TDO) was proposed and applied to cloud computing intrusion detection by C. Kavitha et al. (2023). Hierarchical and Ensemble Approaches: Robson V. Mendonça et al. (2021) proposed Tree-CNN hierarchical algorithm with Soft-Root-Sign (SRS) activation function to enhance depending on any type of attacks and effectiveness. In their studies in 2022, Edeh Michael Onyema et al., ensembled machine learning with biological intelligence within the Cyborg Intelligence framework suggesting increased precision and decreased number of false positives.

Machine Learning and Deep Learning Models

Deep Learning Models: Tongtong Su et al. (2020) have introduced BAT-MC model, which is based on Bidirectional Long Short-term Memory (BLSTM) and attention mechanism for network anomaly detection outperforming other techniques. Alrayes, Fatma S., et al presented an end-to-end CNN network that adds channel attention mechanisms to achieve 99.728% accuracy. Mohammed Zakariah et al. (2023) designed an IDS using an artificial neural network, and it recorded Achieved 97.5% accuracy, whereas KNN, SVM, LSTM, and DNN models was lower. Machine Learning Classifiers: For the NSL-KDD dataset, various classifiers (SVM, KNN, LR, NB, MLP, RF, ETC, DT) are compared by Iram Abrar et al. (2020) of which RF, ETC and DT have more than 99% accuracy. Sarthak Rastogi et al. (2022) have compared SVM, Naive Bayes, KNN, Random Forest, Logistic Regression, Decision Tree using the NSL-KDD dataset for IDS proposing to use KNN & Random Forest algorithms. Carlisle Adams et al. (2020) compared the machine learning algorithms for IoT networks for which XGBoost delivered an accuracy of 97% and 99.6% AUC. Hybrid and Advanced Approaches: Ammar Aldallal et al. (2021) in their work used the integration of SVM with the genetic algorithm as a proposed hybrid IDS that exhibited higher accuracy and performance as compared to benchmark solutions.; IDS were strengthened in 2021 by Subarna Shakya with machine learning integrated to a Grey Wolf Optimization algorithm known as MLGWO; this improved the detection rate and the number of false alarms.

Application-Specific Approaches and Optimizations

IoT and Edge Security: In Ban Salman Shukur et al. (2022) developed the AI-SM-IoT for the security of the edge network with a detection rate of 93.5 % enhancement of the delay and an improvement of packet delivery. IDS applied for RPL based IoT networks was proposed by Faiza Medjek et al. (2021) using Decision Tree, Random Forest and K-Nearest Neighbors with the accuracy over 99% in detecting routing attacks. Joseph Kipongo et al. (2023) have done research on a honeycomb structure-based IDS for SDWSNs with the use of advanced techniques such as TLDQN and Bi-GAN and the proposed model outperformed the existing models in terms of energy consumption, latency and security. Cloud and Wireless Sensor Networks: Lalit Kumar Vashishtha, et al (2023) proposed a cloud hybrid IDS consist of signature-based IDS, anomaly based IDS is used were found fit sense high results in different datasets. Hanaa Attou et al. (2023) presented cloud-based IDS with random forest and feature engineering with 98.3% accuracy for the Bot-IoT dataset and 99.99 with NSL-KDD. Wireless multi-channel network throughput was enhanced using an artificial intelligence algorithm and Artificial Bee Colony (ABC) optimization by Dr. P. Ebby Darney et al. (2021). General Security Enhancements: Inadyuti Dutt et al. (2020) proposed an IDS using the concept of an artificial immune system model for anomaly detection; they conveyed increased true positive values and decreased false positive values. In their work, Abdel-Rahman Al-Ghuwairi et al. (2023) put forward a time series-based IDS based on collaborative features selection and Face book Prophet Selection techniques where they reduced predictors and enhanced performance measures.

Gap Analysis

These literatures indicate useful progress in IDS, with application of machine learning as well as deep learning methods. However, most of the times, methods by Ali et al. (2024) and other hybrid models require additional optimization algorithms that include Artificial Bee Colony (ABC) or Genetic Algorithms (GA). All these optimizations work perfectly but come with added complexities and computation costs. As a result, there is the lack of comparable evaluation of standalone ANN and LSTM models the NSL-KDD dataset without further enhancements. This work will try to fill the gap of evaluating these neural models when used in a simpler manner, thus revealing inherent strengths and weaknesses of these models.

3 Research Methodology

The research approach for this project is based on development, training, and evaluation of Artificial Neural Networks (ANN) and Long Short-Term Memory (LSTM) models for intrusion detection. The research procedure involved several key stages:

Data Collection and Preprocessing:

The primary dataset that has been used is called the NSL-KDD dataset. The data preprocessing includes resampling and oversampling, normalization of features, and informational separation of training samples, validation samples, and test samples were made in the current study.

Data set: The NSL-KDD dataset is obtainable at Kaggle and consists of 41 features per 125,973 records with each record marked as either normal or an attacker engaged in network traffic records. Specifying class imbalance and redundancy as the dominant problems in the original KDD Cup dataset, it is improved.

Data set Link: https://www.kaggle.com/datasets/kiranmahesh/nslkdd?select=kdd_train.csv

Model Development:

Two neural network models were developed for this study:

Artificial Neural Network (ANN): The ANN model was built with several hidden layers. Tuning on such hyperparameters as the number of neurons, activation functions, and dropout rates occurred. Back propagation and gradient descent were used at the training process. It was assessed on the test set in terms of accuracy, precision, recall, and F1-score.

Long Short-Term Memory (LSTM): The LSTM model was built and trained on sequence data learning temporal relations. Thus, other hyperparameters including LSTM units as well as dropout rates were adjusted. In the testing set accuracy, precision, recall, and F1-score were used as the performance metrics was measured.

Training and Validation: All these models were trained on NSL-KDD training subset while the validation set was used for model selection and over fitting prevention. Through back propagation weights were adjusted in an iterative manner while using the Adam optimizer for optimization.

Model Comparison and Selection: Comparison of ANN and LSTM models was made as per accuracy, precision, recall, and F1-score measurements. Out of two models created, the one that had the best performance measure was taken for implementation.

Inference System with IDS Threat Detection and Logging:

The inference system uses the trained IDS models to classify and detect network threats in real-time. Detected threats are logged in a log.csv file, detailing the type of attack. The system flags and reports each threat, providing crucial information for security analysis and response.

Evaluation Methodology

Performance Evaluation:

Accuracy: The term accuracy describes the ratio of all forecasted to the like proportion of accurate, negative or positive ones. Total proportion of correctly classified instances but this is not appropriate if the classes are skewed.

Precision: The way that a binary classifier is measured as to how efficient it is in giving fewer false positive results or in other words how good is the classifier at arriving at the right percentage of the positives that are real. Where the false positive may be financially significant, it is necessary.

Recall: The recall parameter what how many of the true positive the classifier distinguishes. While it may draw in some extra non-positives into the tally, it can really boost the positives when that is the goal.

F1-Score: Often, precision and recall have to work in parallel; in such cases, the F1 factor received with the use of the formula 5 which is the factor of harmonic mean of the precision and recall will be helpful.

Confusion matrix: Confusion matrix is one of the useful tools in the determination of the performance of the algorithms in charge of categorization. It offers a very detailed indication of the distance between the expected class label and the actual one. The elements that make it up are as follows: These are as follows:

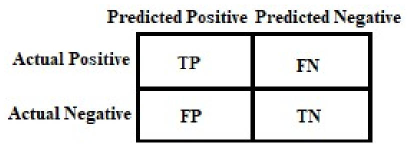
True Positives (TP): The number of points which have been misclassified and actually come under the negative class.

True Negatives (TN): The number of correct classifications made on the negative class.

False Positives (FP): The number of instances, which are identified to be of negative class of the spectrum but actually belongs to the positive class of the spectrum.

False Negatives (FN): The count of samples that are positive belonging to a particular class but have been classified under the negative class.

In figure 1, the confusion matrix is usually displayed as a 2x2 table for binary classification issues.



	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Figure 1: Example Confusion Matrix

Classification matrix: Classification report is an assessment tool that is used in machine learning; it is an elaborated report on performance of any classification algorithm. It finds extensive application in binary and multiclass problems.

Prototype Development: This is to construct an Inference system with the interconnection of the best performing IDS model and this Inference system is checked with the new sample IOT transaction data and only checked whether this Inference System is working or not.

This approach of evaluating the models was used to make sure that the models were subjected to a lot of tests similar to the actual environment. This included:

- **Accuracy and Precision Analysis:** Predictive ability was measured based on the models' ability to accurately classify and localize intrusions with specific reference to false alarms and their perils to security.
- **Comparative Performance Analysis:** The two models that were used were the ANN and LSTM in order to compare the tradeoff between accuracy of inference and detection of pattern complexity.

Tools and Equipment

The following tools and equipment were utilized in the research:

Software Requirements: Windows 10 (64-bit), Python 3, Anaconda, Flask, Keras, TensorFlow, OpenCV, Matplotlib, Scikit-learn, Numpy, Pandas, Jupyter Notebook (with Notepad++ Editor, Anaconda Navigator, and tools for data visualization and machine learning).

Hardware Specifications: Intel Core i7, 1TB - hard disk, 12GB - RAM, high-performance computing resources (GPUs, cloud-based options recommended for deep learning model training).

4 Design Specification

Referring to the system architecture presented in Figure 2, it is possible to point out that the proposed system would cover all stages that are characteristic for the machine learning model development: data cleaning and preprocessing; feature engineering; training of the model; and finally model inference. It is designed to support the training and the input stages of such models so that they can be learned and then used for predictions on the fly. On the training side, the process starts with NSL-KDD dataset which is one of the most used datasets in the network intrusion detection. First, the raw data is preprocessed to extract the features out of it. This involves implementing of categorical data which involves converting them into numerical form, scaling of features which involve putting all features to the same scale, dimensionality reduction which involves choosing the most relevant features to boost the model's performance. Following data preprocessing the dataset is then split into a training set and a set for validation and testing respectively. This division makes it possible to expose the models to the maximum possible number of situations during training and to evaluate their work in all directions. The second stage in the proposed model is the model training stage and this involves training the model using either An Artificial Neural Network (ANN) or Long Short Term Memory (LSTM) network. These models are built by feeding the preprocessed training data to it and training takes place in such a way that the weight of the model is adjusted based on the back propagation and the optimization algorithms like the Adam optimizer. The trained models are then tested using the validation set so as to validate these models. The validation phase yields metrics such as the accuracy score, a classification report, and confusion matrix. All of them are used to establish the extent to which the model has learned from data. These metrics are useful in deciphering the strengths and the areas of the model that might need further tweaking in case of a need.

On the user side, the architecture is built load-balance for real-time inference. As the following input from users, which could be new traffic pattern or other related signals needed to be analyzed, user input data is pre-processed to be in the same format as the model used in the previous training. This may include steps such as feature extraction, and features which are selected have to match the features used during the model building phase. After obtaining the above data, the data is used to make a prediction in the model. Based on the findings of the model, predictions are produced; threats are detected in the network traffic or normal activity. The predictions made are checked then logged with the outcome of the results. This system architecture is therefore a modular and most importantly scalable, processes necessitated for accomplishing machine learning from data preprocessing, model training, validation and finally the prediction phase. The architecture also provides scalability, it may help accommodate larger data sets and more complex models allow for effective intrusion detection in evolving network.

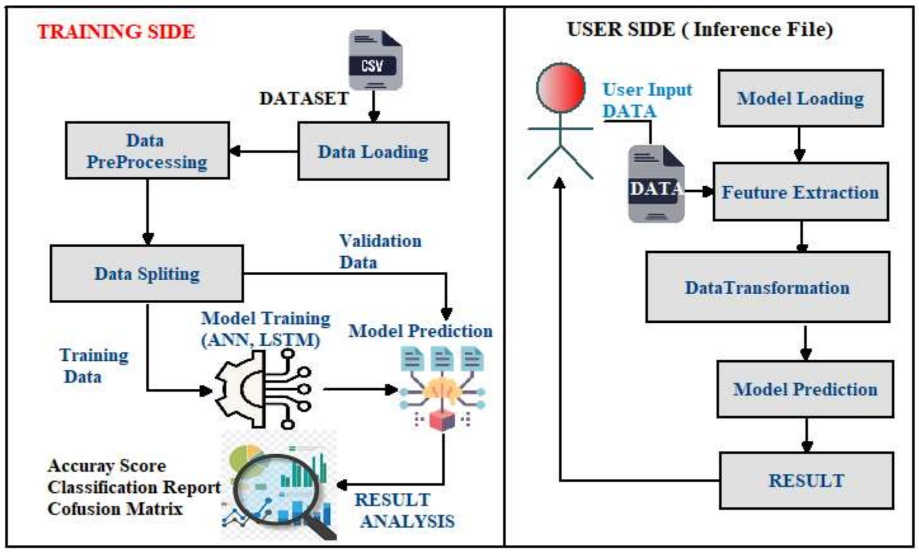


Figure 2: System architecture

5 Implementation

Data Preprocessing: The implementation started with data preprocessing mainly in which the NSL-KDD dataset underwent a data cleaning, normalization and transformation process. This involved feature scaling where categorical features were encoded as numerical features and pre-processing of the data was done by normalizing it. The dataset was then partitioned into training, development/validation, and test datasets so as to avoid overfitting of the models. Key attributes in the dataset included timestamps, protocol types, service types, flags, and byte counts, with specific focus on host-based traffic patterns such as 'dst_host_srv_diff_srv_rate' and 'dst_host_same_src_port_rate'.

Data Transformation and Visualization: Data transformations were applied to prepare the dataset for modeling. Types of charts to capture the pattern of the feature and illustrate their relationship were employed to choose features: degrees of the network connection flag (Figure 3). Such steps were quite helpful in pre-processing the data for model training.

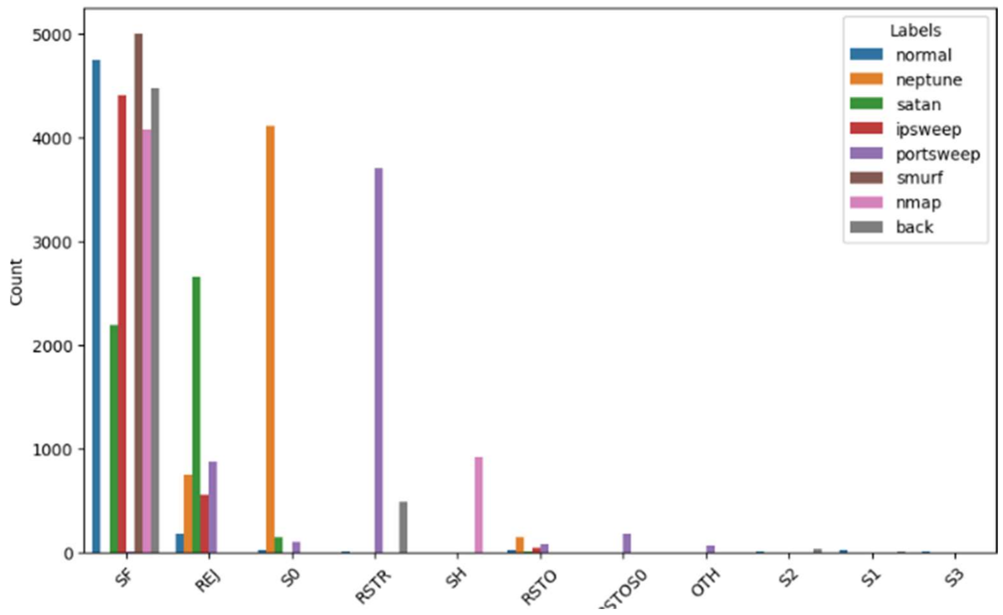


Figure 3: Connection flags across various traffic labels

Model Development: When it comes to the implementation it was emphasized on creating and training two machine learning models: Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) network. The ANN consisted of several layers, and the LSTM model was aimed at addressing temporal dependencies in network traffic. The models were trained on preprocessed and fine-tuned on the certain validation set.

Training and Evaluation: Training was followed by examination of models' performance on a test set in terms of metrics like accuracy, precision of the model, recall and F1 score. Figure 4 demonstrates that while both the ANN and LSTM classifiers were used to predict attacks based on the ratio of the number of successful attacks to executed actions and on the 'average number of attempts per successful attack', respectively, the LSTM was more precise and had better recall as compared to the ANN for both frequent and rare attacks. Using confusion matrix, it was observed that LSTM can detect subtle attack patterns.

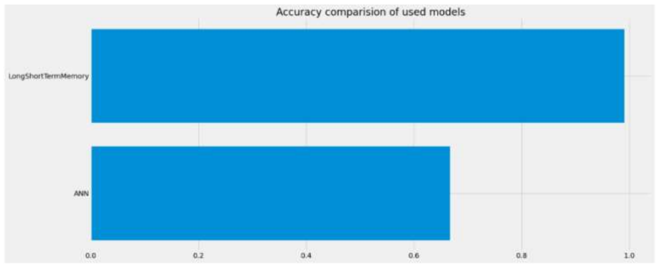


Figure 4: Comparing between ANN& LSTM

Final Outputs and Model Deployment

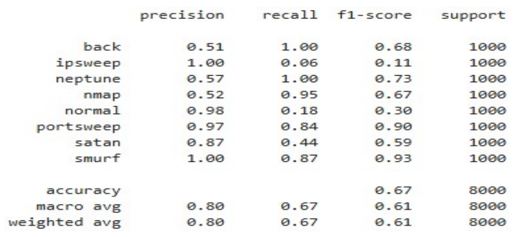
The last phase of the proposed implementation included using the trained models in an actual time intrusion detection system. The system was intended to store predictions into a file which presents the attack that has been identified, and the time that this attack had happened. The models were then tested where they were able to identify different types of network intrusions showing the efficiency of model development stages.

This implementation process from data preprocessing step all the way to deployment staged built accurate models for network intrusion detection. Selecting the right features for data transformation and implementing models had a great impact and testing of the model with an inference system for IDS perdition, by Evaluation of the models to determine the best performing for inference system.

6 Evaluation

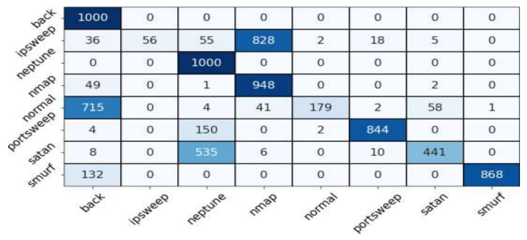
Artificial Neural Network

Performance Overview: The ANN algorithm gives an overall accuracy of 66.70 %, precision; recall, and f1-scores varies from 0.51 to 1.00. The smurf class emerged as the best performing class while the back class had the best recall. In the plot of accuracy of the ANN model, there is a gradual increase in the accuracy, however, there are fluctuations ideally it should have been a perfectly straight line. In loss plot, it can also be observed for signs of overfitting in the form of huge jumps of the validation loss toward the end of the epochs.



```
              precision    recall  f1-score   support

        back       0.51      1.00      0.68      1000
     ipsweep       1.00      0.06      0.11      1000
     neptune       0.57      1.00      0.73      1000
        nmap       0.52      0.95      0.67      1000
      normal       0.98      0.18      0.30      1000
   portsweep       0.97      0.84      0.90      1000
       satan       0.87      0.44      0.59      1000
       smurf       1.00      0.87      0.93      1000

    accuracy                           0.67      8000
   macro avg       0.80      0.67      0.61      8000
weighted avg       0.80      0.67      0.61      8000
```

Figure 5: Classification report for ANN



	back	ipsweep	neptune	nmap	normal	portsweep	satan	smurf
back	1000	0	0	0	0	0	0	0
ipsweep	36	56	55	828	2	18	5	0
neptune	0	0	1000	0	0	0	0	0
nmap	49	0	1	948	0	0	2	0
normal	715	0	4	41	179	2	58	1
portsweep	4	0	150	0	2	844	0	0
satan	8	0	535	6	0	10	441	0
smurf	132	0	0	0	0	0	0	868

Figure 6: Confusion matrix for ANN

Classification Report: Figure 5 with Classification Report metrics precision, recall, f1-score and support. The classes include back, ipsweep, neptune, normal, portsweep, satan and smurf with each of them featuring 1000 instances except the normal class that has 8000. The precision, recall as well as the f1-score vary between 0.51 to 1.00, again the highest precision and the f1-score for the smurf class and the highest recall for the back class. The overall accuracy is 0.67 respectively and the macro and weighted average f1 scores are 0.61.

Confusion matrix: The following figure 6, a confusion matrix provides information regarding the effectiveness of a classification algorithm. The entries along the diagonal are the number of correct classifications, and the entries off the diagonal are the misclassifications. The number of correct classifications for each class is as follows: they have been computed with the following values: : back (1000), lipsweep (36), neptune (0), nmap (49), normal (715), portsweep (4), satan (8), and smurf (868). Misclassifications include back (0), lipsweep (55), neptune (0), nmap (1), normal (41), portsweep (150), satan (0), and smurf (0).Total accuracy is calculated as 3429 out of 3429.

Accuracy plot for ANN: The Accuracy plot of ANN model over 8 epochs is shown in Figure 7. It shows a line graph with the training and validation accuracy of ANN model for 20 epochs. The training accuracy in blue rises slightly to below 0.9. The validation rate, depicted by the red line, increases but with oscillations getting slightly higher than 0.8 in Epochs 5 it notable dips and at 17.5 it spikes in between there is oscillation due to dataset.

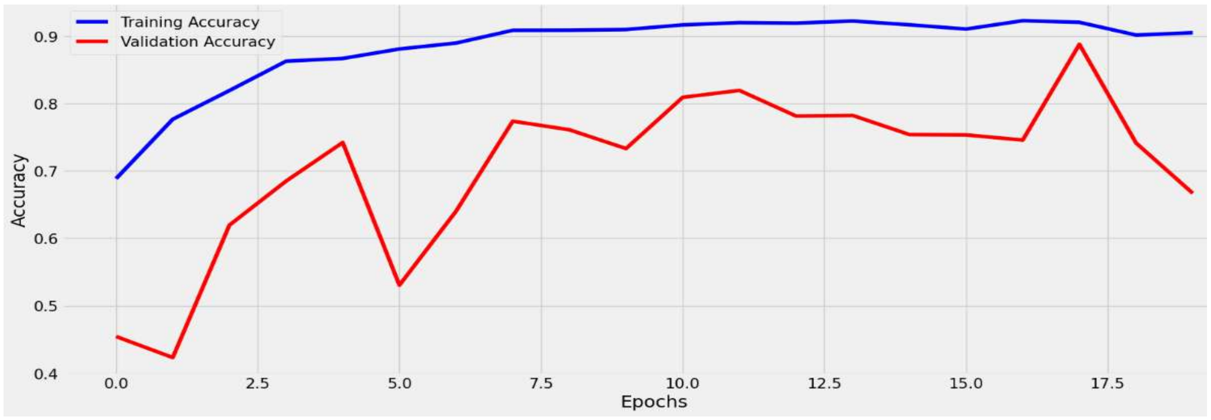


Figure 7: Accuracy plot for ANN

Loss Plot for ANN: The graph presented in figure 8 is known as the Loss plot of ANN model. Training loss is shown as blue line while validation loss is shown as red line in the figure above. The plot below shows changes in the training and validation losses: sometimes the validation loss sharply increases that can mean that the ANN overfitting or instability in the data.

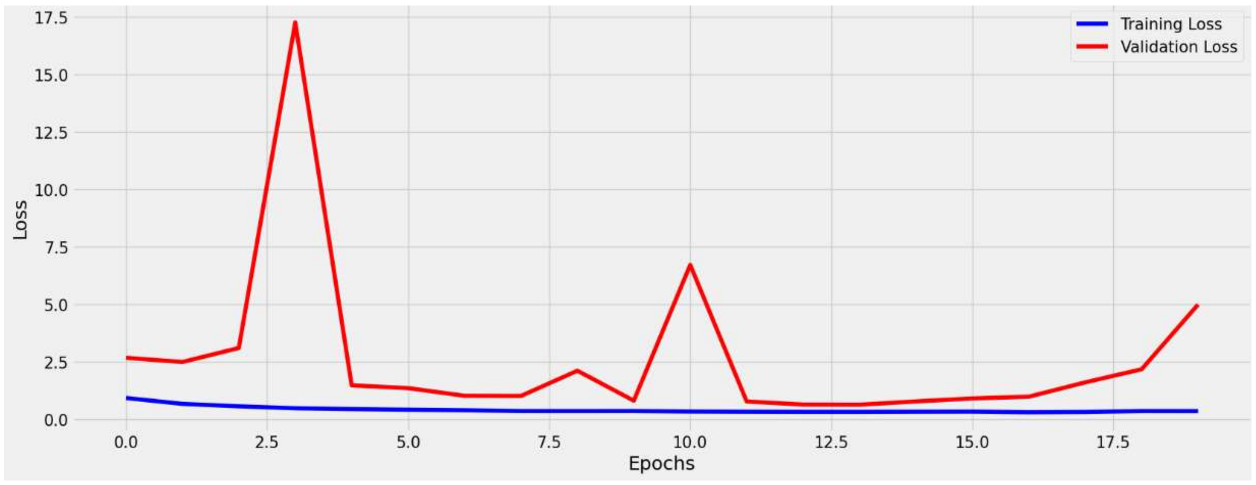
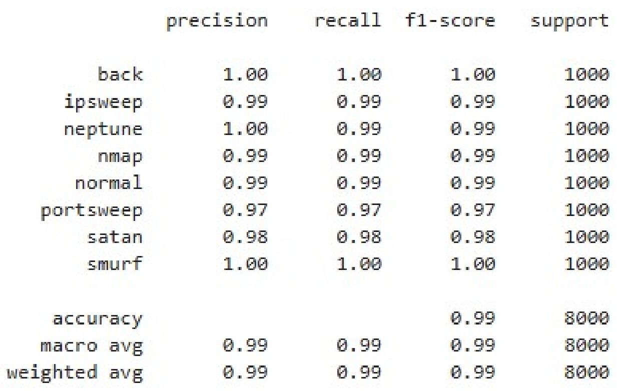


Figure 8: Loss plot for ANN

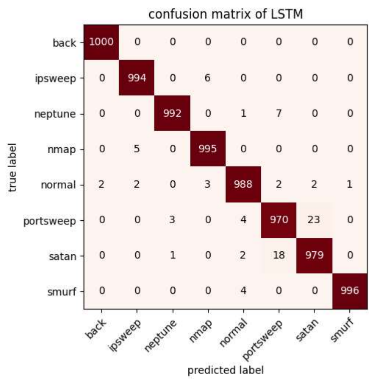
Long Short Term Memory Classifier (LSTM) Evaluation

Performance Overview: The LSTM model is well performed up to the mark and it achieves the classification accuracy of 99.10%. The precision, recall, and F1-scores are high for all the categories, the lowest being 0.98 to 1.00, which is said to represent the model's capacity to diagnose network intrusions. The confusion matrix again supports this with very low levels of misclassification and a highly diagonal distribution which shows the model's effectiveness in identifying a range of different attack types.



```
              precision    recall  f1-score   support

        back       1.00      1.00      1.00      1000
     ipsweep       0.99      0.99      0.99      1000
     neptune       1.00      0.99      0.99      1000
        nmap       0.99      0.99      0.99      1000
      normal       0.99      0.99      0.99      1000
   portsweep       0.97      0.97      0.97      1000
       satan       0.98      0.98      0.98      1000
       smurf       1.00      1.00      1.00      1000

    accuracy                           0.99      8000
   macro avg       0.99      0.99      0.99      8000
weighted avg       0.99      0.99      0.99      8000
```

Figure 9: Classification report for LSTM



	back	ipsweep	neptune	nmap	normal	portsweep	satan	smurf
back	1000	0	0	0	0	0	0	0
ipsweep	0	994	0	6	0	0	0	0
neptune	0	0	992	0	1	7	0	0
nmap	0	5	0	995	0	0	0	0
normal	2	2	0	3	988	2	2	1
portsweep	0	0	3	0	4	970	23	0
satan	0	0	1	0	2	18	979	0
smurf	0	0	0	0	4	0	0	996

Figure 10: Confusion matrix for LSTM

Classification Report: Figure 9 shows quantitative which therefore take values from the range 0.98 to 1.00 denoting rather high values of the performance indicators. Refer again to the figure; the 'support' column features serial numbers that seem to indicate counts – 1000 and 8000. The 'accuracy', the 'macro avg', and the 'weighted avg' rows show macro-averages; the 'accuracy' is 0.99 'weighted avg' also being 0.99. Such metrics imply a high performance of the model or system under consideration in terms of accuracy and roughly equal performance in all categories.

Confusion matrix: A detailed confusion matrix is given in Figure 10 that shows the prediction accuracy of an LSTM (Long Short-Term Memory) neural network. The rows contain the true labels, and the columns contain the predicted labels. The diagonal cells: from top left to bottom right, show the number of instances in classes that have been classified correctly. For instance, in the cell at the cross-section of the 'back' row and the 'back' column there is the figure 1000 which represents the number of all instances of the 'back' type that were classified correctly. The off-diagonal cells reveal the misclassifications; for example, six 'ipsweep' data points were classified as 'back', and twenty-three 'normal' data points as 'portsweep'. In the case of the model, the degree of accuracy ranges to about 99.8% can be deduced from diagonal values which are high, whereas the off-diagonal values which are low. The total numbers of actual instances for train are 7893, which is the overall output that has been obtained at the end of the model for diagonal values.

Accuracy plot for LSTM: Figure 11 demonstrates a line graph with a blue line and an orange line; both lines present two various metrics in a spectrum of epochs. The blue line, which represents 'Accuracy', begins at roughly 0.65 at epoch 0, and increases over time and gets slightly above 1 at epoch 17.5. This orange line called 'Validation accuracy' begins slightly above 0.60 at epoch 0 and keeps increasing up to 17.5 epoch at below 1. The curves are virtually parallel to each other, which means that the increase or rather the decrease in both parameters is simultaneous across the epochs.

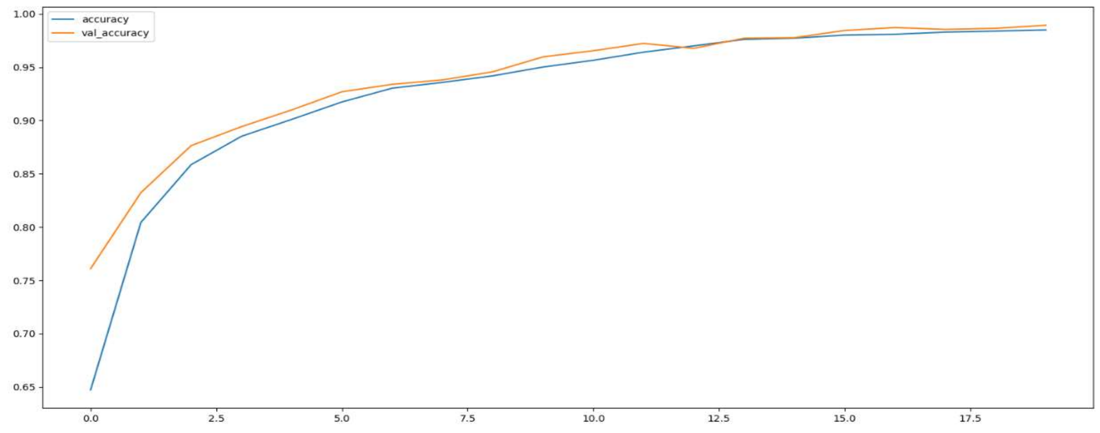


Figure 11: Accuracy plot graph for LSTM

Loss Plot for LSTM: Figure 12 is a loss plot graph with two lines: The blue line which represents the 'loss' and the other in orange show the 'Validation Loss'. At the time of the initial training, the 'loss' stands at about 1.6 and the 'Validation Loss' is approximately 1.4. As can be observed, as the number of epochs rises, values of both 'loss' and 'Validation Loss' become lower: the 'loss' is below 0.2 by the end of the training and 'Validation Loss' a little above 0.2. This can be considered as a sign of overfitting, hence the relatively small difference between the training loss and the validation loss. As can be seen from the plot both lines are going down towards the origin, which means that the error of the model is going down with time, and thus it shows successful learning.

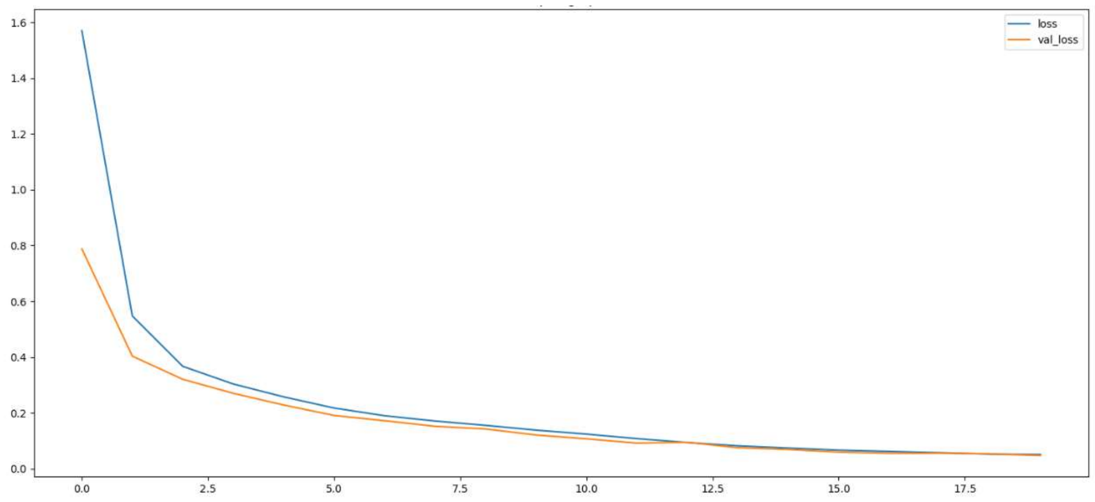
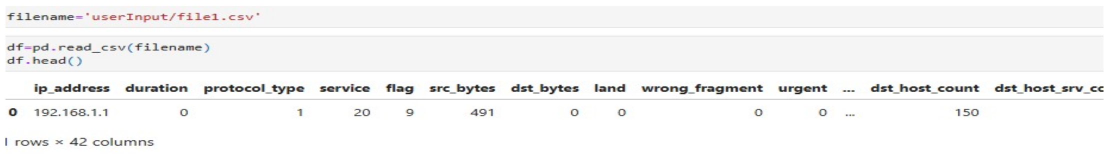


Figure 12: Loss plot graph for LSTM

Inference

The inference process uses data from a CSV file, a pre-trained LSTM model that can identify network threats. First the script checks if the IP address is included in the existing blocklist which is Block_IP_List.csv. If the given IP address does not fall in the block list, the data is given to the LSTM model to predict if the traffic is malicious or normal. In case of an attack prognostic, the IP is then blocked so as not to expect other threats from the same IP. If the prediction is normal, the IP is genuine and nothing is done, no action is further pursued. This way it reduces threats and allows for regular updates on the block list depending on newly discovered IPs with malicious intent. The inference script does the threat detection part with the help of data taken from a CSV file, with the help of pre trained LSTM model and the login the result. It performs the following steps:

Reads input data from a CSV file named as (userInput/file1.csv) into a DataFrame as shown in Figure 13.



```
filename='userInput/file1.csv'
```

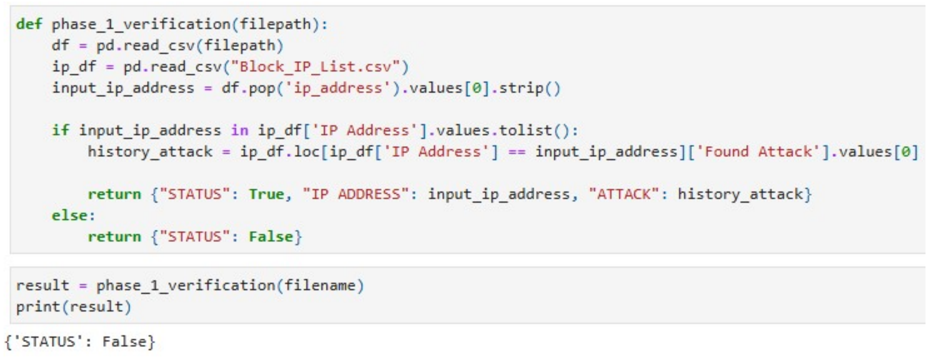
```
df=pd.read_csv(filename)
df.head()
```

	ip_address	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	...	dst_host_count	dst_host_srv_cc
0	192.168.1.1	0	1	20	9	491	0	0	0	0	...	150	

I rows × 42 columns

Figure 13: Data Loading Visualization

Checks if the input IP address exists in file names (Block_IP_List.csv), if found stops processing. As shown in Figure 14 the input IP is not in Blacklist, so the file moves forward, this happened due to the Block_IP_List.csv being fresh without any Block listed IP's.



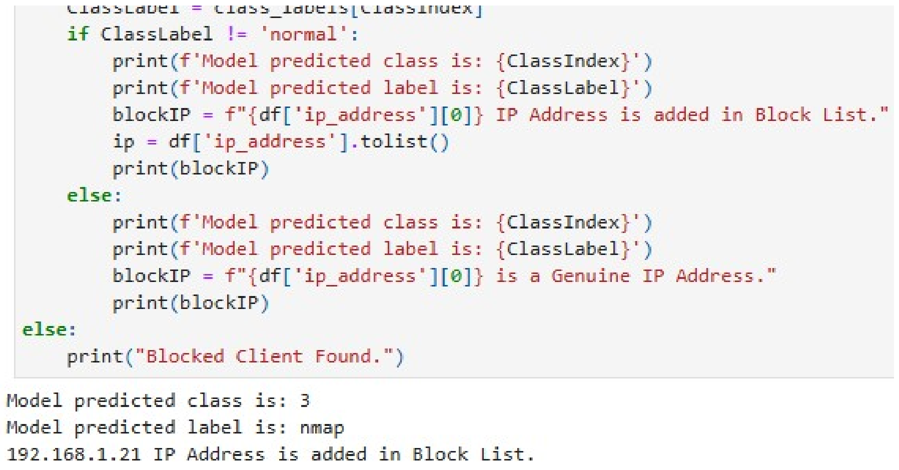
```
def phase_1_verification(filepath):
    df = pd.read_csv(filepath)
    ip_df = pd.read_csv("Block_IP_List.csv")
    input_ip_address = df.pop('ip_address').values[0].strip()

    if input_ip_address in ip_df['IP Address'].values.tolist():
        history_attack = ip_df.loc[ip_df['IP Address'] == input_ip_address]['Found Attack'].values[0]

        return {"STATUS": True, "IP ADDRESS": input_ip_address, "ATTACK": history_attack}
    else:
        return {"STATUS": False}
```

```
result = phase_1_verification(filename)
print(result)
```

```
{'STATUS': False}
```

Figure 14: IP Block list Check

Pre trained model (LSTM), predicts the class label, and determines if the traffic is malicious. Figure 15 shows that the model predated it as attack and sends IP to be blacklisted.

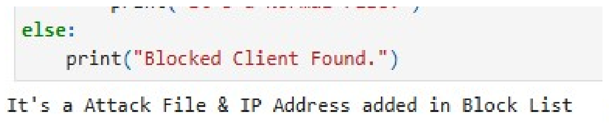


```
    ClassLabel = class_labels[ClassIndex]
    if ClassLabel != 'normal':
        print(f'Model predicted class is: {ClassIndex}')
        print(f'Model predicted label is: {ClassLabel}')
        blockIP = f"{df['ip_address'][0]} IP Address is added in Block List."
        ip = df['ip_address'].tolist()
        print(blockIP)
    else:
        print(f'Model predicted class is: {ClassIndex}')
        print(f'Model predicted label is: {ClassLabel}')
        blockIP = f"{df['ip_address'][0]} is a Genuine IP Address."
        print(blockIP)
else:
    print("Blocked Client Found.")
```

```
Model predicted class is: 3
Model predicted label is: nmap
192.168.1.21 IP Address is added in Block List.
```

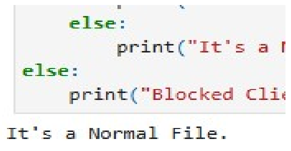
Figure 15: Prediction Workflow

The script adds the IP address to Block_IP_List.csv if an attack is detected as shown in Figure 16. if there is no attack detected it shows "It's a Normal File" as shown in Figure 17.



```
else:
    print("Blocked Client Found.")
```

```
It's a Attack File & IP Address added in Block List
```



```
    else:
        print("It's a 
else:
    print("Blocked Cli
```

```
It's a Normal File.
```

Figure 16: Blacklist Update **Figure 17: A Normal File**

The entire workflow ensures efficient threat detection, logging, and blocking, making it a comprehensive solution for network security monitoring.

6 Discussion

The studies showed that the ANN model optimized to validation accuracy of 82% using 25 epoch number but cannot provide the best accuracy beyond those and need the help of some optimization techniques like Artificial Bee Colony (ABC). On the other hand, LSTM learning models had high accuracy with 99% validation accuracy within the first 20 epochs. It does not need any special optimized algorithm. This distinct variation shows that LSTM has tools for dealing with the features of NSL-KDD datasets better than ANN when using a lesser number of epochs for training than ANN. Established an inference model which used LSTM as a base that will enable initial recognition of data packets in this prototype model. More importantly, if an intrusion is identified in the system, then this model adds the culprit's IP address to its blacklist to improve its defense mechanisms to promptly deal with the threat as soon as it is detected.

7 Conclusion

The studies conducted in the project show the results obtained bring into light the merits and demerit of different models of neural network in relation to intrusion detection. ANN though can perform well, the performance on the high dimensional data set like NSL-KDD is quite moderate if no extra optimization is made. However, even fewer epochs were showing high accuracy of Long Short-Term Memory (LSTM) models when it came to managing complicated data sets. From the comparison of the resultant tables, it could be deduced that LSTM is more appropriate where accuracy and efficiency are very crucial in a certain task. The use and analysis of an LSTM-based inference model that is capable of real-time detection of intrusion and also the blocking of malicious IP address makes the case study very practical for boosting up the security of a networks.

Further research work will look at how other machine learning algorithms like CNN as well as combined architectures would help in refining the intrusion detection system. It should also be incorporated to such procedures as Particle Swarm Optimization (PSO), and Genetic Algorithms in the future to increase the efficiency of developed models. Moreover, it is the plan to compare the results of these models with different datasets of IDS and to use PySpark for big data stream processing in real-world networks environments.

References

Ali, M., Razaque, A., Yeskendirova, D.M., Nurlybayev, T.A., Askarbekova, N.Y. and Kashaganova, Z.A., 2024. Enhancing cloud computing security through AI-based intelligent intrusion detection leveraging neural networks and artificial bee colony optimization.

International Information Technology University. Available at: [URL if available] [Accessed 9 August 2024].

Abrar, I., Ayub, Z., Masoodi, F. and Bamhdi, A.M., 2020, September. A machine learning approach for intrusion detection system on NSL-KDD dataset. In *2020 international conference on smart electronics and communication (ICOSEC)* (pp. 919-924). IEEE.

Aldallal, A. and Alisa, F., 2021. Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry*, *13*(12), p.2306.

Al-Ghuwairi, A.R., Sharrab, Y., Al-Fraihat, D., AlElaimat, M., Alsarhan, A. and Algarni, A., 2023. Intrusion detection in cloud computing based on time series anomalies utilizing machine learning. *Journal of Cloud Computing*, *12*(1), p.127.

Alrayes, F.S., Zakariah, M., Amin, S.U., Khan, Z.I. and Alqurni, J.S., 2024. CNN Channel Attention Intrusion Detection System Using NSL-KDD Dataset. *Computers, Materials & Continua*, *79*(3).

Attou, H., Guezzaz, A., Benkirane, S., Azrour, M. and Farhaoui, Y., 2023. Cloud-based intrusion detection approach using machine learning techniques. Big Data Mining and Analytics, 6(3), pp.311-320.

Davahli, A., Shamsi, M. and Abaei, G., 2020. A lightweight Anomaly detection model using SVM for WSNs in IoT through a hybrid feature selection algorithm based on GA and GWO. *Journal of Computing and Security*, *7*(1), pp.63-79.

Dutt, I., Borah, S. and Maitra, I.K., 2020. Immune system based intrusion detection system (IS-IDS): A proposed model. *IEEE Access*, *8*, pp.34929-34941.

Fatani, A., Abd Elaziz, M., Dahou, A., Al-Qaness, M.A. and Lu, S., 2021. IoT intrusion detection system using deep learning and enhanced transient search optimization. *IEEE Access*, *9*, pp.123448-123464.

Haddad, N.M., Salih, H.S., Shukur, B.S., Abd, S.K., Ali, M.H. and Malik, R.Q., 2022. Managing Security in IoT by Applying the Deep Neural Network-Based Security Framework. *Eastern-European Journal of Enterprise Technologies*, *6*(9), p.120.

Jacob, I.J. and Darney, P.E., 2021. Artificial bee colony optimization algorithm for enhancing routing in wireless networks. *Journal of Artificial Intelligence*, *3*(01), pp.62-71.

Kavitha, C., Gadekallu, T.R., K, N., Kavin, B.P. and Lai, W.C., 2023. Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing. Electronics, 12(3), p.556.

Kavitha, S. and Uma Maheswari, N., 2021. Network anomaly detection for NSL-KDD dataset using deep learning. *Information Technology in Industry*, *9*(2), pp.821-827.

Kipongo, J., Swart, T.G. and Esenogho, E., 2023. Artificial Intelligence-Based Intrusion Detection and Prevention in Edge-Assisted SDWSN With Modified Honeycomb Structure. *IEEE access*.

Kumar, P., Kumar, A.A., Sahayakingsly, C. and Udayakumar, A., 2021. Analysis of intrusion detection in cyber attacks using DEEP learning neural networks. Peer-to-Peer Networking and Applications, 14(4), pp.2565-2584.

Liu, J., Kantarci, B. and Adams, C., 2020, July. Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In *Proceedings of the 2nd ACM workshop on wireless security and machine learning* (pp. 25-30).

Medjek, F., Tandjaoui, D., Djedjig, N. and Romdhani, I., 2021. Fault-tolerant AI-driven intrusion detection system for the internet of things. *International Journal of Critical Infrastructure Protection*, *34*, p.100436.

Mendonça, R.V., Teodoro, A.A., Rosa, R.L., Saadi, M., Melgarejo, D.C., Nardelli, P.H. and Rodríguez, D.Z., 2021. Intrusion detection system based on fast hierarchical deep convolutional neural network. IEEE Access, 9, pp.61024-61034.

Mohammed, B. and Gbashi, E.K., 2021. Intrusion detection system for NSL-KDD dataset based on deep learning and recursive feature elimination. *Engineering and Technology Journal*, *39*(7), pp.1069-1079.

Ngueajio, M.K., Washington, G., Rawat, D.B. and Ngueabou, Y., 2022, September. Intrusion detection systems using support vector machines on the kddcup'99 and nsl-kdd datasets: A comprehensive survey. In *Proceedings of SAI Intelligent Systems Conference* (pp. 609-629). Cham: Springer International Publishing.

Nizamudeen, S.M.T., 2023. Intelligent intrusion detection framework for multi-clouds–IoT environment using swarm-based deep learning classifier. *Journal of Cloud Computing*, *12*(1), p.134.

Onyema, E.M., Dalal, S., Romero, C.A.T., Seth, B., Young, P. and Wajid, M.A., 2022. Design of intrusion detection system based on cyborg intelligence for security of cloud network traffic of smart cities. *Journal of Cloud Computing*, *11*(1), p.26.

Rastogi, S., Shrotriya, A., Singh, M.K. and Potukuchi, R.V., 2022. An analysis of intrusion detection classification using supervised machine learning algorithms on nsl-kdd dataset. *Journal of Computing Research and Innovation*, *7*(1), pp.124-137.

Shakya, S., 2021. Modified gray wolf feature selection and machine learning classification for wireless sensor network intrusion detection. *IRO Journal on Sustainable Wireless Systems*, *3*(2), pp.118-127.

Su, T., Sun, H., Zhu, J., Wang, S. and Li, Y., 2020. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access*, *8*, pp.29575-29585.

Vashishtha, L.K., Singh, A.P. and Chatterjee, K., 2023. HIDM: A hybrid intrusion detection model for cloud based systems. *Wireless Personal Communications*, *128*(4), pp.2637-2666.

Zakariah, M., AlQahtani, S.A., Alawwad, A.M. and Alotaibi, A.A., 2023. Intrusion Detection System with Customized Machine Learning Techniques for NSL-KDD Dataset. *Computers, Materials & Continua*, *77*(3).