The International Innovations Journal of Applied Science (IIJAS) Vol. 1, No.2, 15-09-2024
المجلد الأول العدد الثاني 15-9-2024 (IIJAS) مجلة ابتكارات الدولية للعلوم التطبيقية
https://doi.org/10.61856/78jczn59

SCIENTIFIC EVENTS GATE

The International Innovations Journal of Applied Science

Journal homepage: https://iijas.eventsgate.org/iijas

ISSN: 3009-1853 (Online)

AI-Driven Security Paradigms: Elevating Cloud Protection with Machine Learning

Bushra Majeed Muter[1*] and Ayat Jasim Mohamed[2]

[1] Department of Specialist Supervision in Wasit Education, Iraq.
[2] Al-Amarah University College, Department: Medical instrumentation Techniques, Iraq.

ARTICLE INFO

Article history:
Received 19 Jul. 2024,
Revised 10 Aug.2024,
Accepted 16 Aug. 2024,
Available online 15 Sep. 2024

Keywords:
AI-driven security
LSTM networks
Network intrusion detection
Behavior-Centric Cybersecurity Center
BCCC dataset, Cybersecurity

ABSTRACT

In the literature, some studies have explored classifying network traffic using Long Short-Term Memory (LSTM) networks to enhance cloud security. We analyzed a dataset—BCCC—that includes various types of network traffic: Benign, Benign-Email-Receive, Benign-Email-Send, Benign-FTP, Benign-SSH, Benign-Systemic, Benign-Telnet, and Benign-Web_Browsing_HTTP-S. Key features examined include fwd_ack_flag_percentage_in_fwd_packets,fwd_ack_flag_percentage_in_total, min_ fwd_header_bytes_delta_len, and handshake_duration. The model performed well in detecting the Benign class, but some classes with fewer samples, such as Benign-FTP and Benign-Email-Receive, require improved precision and recall due to class imbalance. Overall, the model's performance in classifying network traffic is strong. This research outlines strategies for addressing class imbalance and refining feature engineering. It provides a foundation for further, more detailed investigations into AI approaches for network traffic classification, highlighting the importance of sample balancing to achieve high accuracy.

1. Introduction

The need for strong cybersecurity is more critical now than ever in this digital world. The more companies rely on cloud services, the more challenging it becomes to protect these systems from cyber threats. Cyber attacks are developing and turning out to be more sophisticated and harder to trace. This, therefore, raises the stakes for the security measures (Maalem Lahcen et al., 2020).

Artificial Intelligence and machine learning have become very helpful in this fight against cyber threats. Among the methods for this topic, one should mention Long Short-Term Memory techniques. LSTMs show the best performance on RNN techniques for tasks that involve only sequential data. This feature allows these networks to be more effective at various applications, including network intrusion detection.

This paper discusses the application of an LSTM network in improving cybersecurity in cloud infrastructures. In this work, we make use of the dataset obtained from the Behaviour-Centric Cybersecurity Center, which has raw network traffic data that will be used in training an LSTM-based intrusion detection system capable of detecting different kinds of cyber threats.

Our approach focuses on understanding the patterns in network traffic overtime. Such an approach allows the model to recognize activities that are unusual compared to the majority, for example, denial-of-service attacks or unauthorized access attempts. We carefully preprocess and analyze the data to prepare it for training the LSTM model. Aleesa, A. M., et al in 2020).

It uses several standard metrics for the evaluation of model performance, including Accuracy, Precision, Recall, and F1-score. We can observe from the results that an LSTM-based IDS is able to detect a wide range of cyber threats; therefore, it has huge potential in the cybersecurity domain.

* Corresponding author.
E-mail address: bushramajeed1975@gmail.com

This work is licensed under a Creative Commons Attribution 4.0 International License.

This work adds up to the already existing volume of works on AI-infused security by providing a means of showing how LSTM networks can contribute to augmenting protection in digital infrastructures. Our findings underline the importance of proactive threat detection within an increasing digital connectivity.

2. Related Work

Arshad Hussain and Ghulam Shabir (Hussain & Shabir, 2024) focus on how AI is changing security practices in a DevSecOps framework. They underline the fact that using machine learning, organizations can manage security effectively at every stage of software creation: from writing a code to deployment and maintenance of applications.

AI algorithms contribute to the discovery of security vulnerabilities rather early. The analysis of large amounts of data in order to detect threats, foresee risks, and automate the responses quickens detection and response times to security issues.

The focus of the authors is on how AI not only automates routine security tasks but also enhances human expertise, which enables teams to focus на more strategic goals while AI does recurring tasks efficiently. Besides, AI gives insights that help organizations prioritize security efforts and use resources where they are really needed. It promotes improvement by learning from earlier incidents and adapting to new threats in real-time.

AI's integration into DevSecOps is that serious security practice change to support organizations' digital asset protection and be a step ahead from potential adversaries.

Phani Sekhar Emmanni (Emmanni, 2024) articulates an argument about the importance of Hybrid Cloud systems toward Enterprise IT and the increasing intricacy of cyber threats against them. Traditional methods of security are hardly equipped to deal with such evolving threats.

The author elaborates on how Artificial Intelligence and Machine Learning capabilities take threat detection and response further in hybrid cloud environments. With AI and ML, like anomaly detection, pattern recognition, and predictive analytics, organizations empower their security strategies to be proactive and adaptable.

This research evaluates the efficiency of AI and ML technologies in detecting and managing security risks against traditional security measures. It also looks at how those technologies can be integrated into existing hybrid cloud settings. The article identifies, based on current practice and relevant case studies, best practice, challenges, and future directions for the use of AI and ML in hybrid cloud security. Ultimately, it serves to underscore how these cutting-edge technologies would help ensure resilience and security for hybrid cloud systems in the face of increasing cyber threats, while at the same time sharing valuable insights into the creation of more secure, resilient digital infrastructures.Iqbal H. Sarker and Md Hasan Furhad (Sarker et al., 2021) speak about the role of Artificial Intelligence in enhancing Cybersecurity, placing this specifically within the Fourth Industrial Revolution—otherwise referred to as Industry 4.0. They state that AI is able to protect any system connected to the Internet from a lot of cyber threats, such as attacks and unauthorized access. The authors consider a few techniques of AI methods: machine learning and deep learning, natural language processing, and rule-based expert systems. These techniques offer solutions for the complex problems in today's cybersecurity by automating security processes to levels that traditional systems cannot reach.

Their paper presents a good overview of "AI-driven Cybersecurity," in which the authors focus on how these AI methods could eventually permit services and management of cybersecurity to be more intelligent. They then present some ensuing research directions to guide future studies in the area. The ultimate goal of this paper is to provide insights and guidelines from the intelligent computing perspective, seeking to enable follow-up studies on all such research interests in this area.

Yongjun Xu and Xin Liu (Xu et al., 2021) present an overview of Artificial Intelligence and machine learning impacting a variety of

https://doi.org/10.61856/78jczn59

subjects belonging to Science, Technology, and daily life. It has been highlighted that Machine Learning techniques are focused on the analysis of huge data sets, extraction of knowledge from them, classification of information based on that knowledge, prediction of the outcome, and promotion of evidence-driven decision making. This capability leads to a range of new innovations and hence acts as a catalyst in the ever-continuing evolution process of AI. The authors review the development of Artificial Intelligence and its prospective applications to central sciences such as information science, mathematics, medical science, materials science, geoscience, life science, physics, and chemistry. They discuss challenges raised by each area of science and how AI techniques might be used to meet these challenges.

It also identifies other emerging research trends that are oriented toward the integration of AI in different scientific fields. In this regard, the paper is purposed to serve as an all-comprehensive guide to research into the potential of AI in the basic sciences, thus nudging the researchers to achieve an understanding of the present applications of AI and promoting their relentless evolution in these disciplines.

3. Proposed Methodology

In this paper, an effort has been made to enhance cybersecurity using Long Short-Term Memory (LSTM) networks. The methodology applied here consists of several steps, which start from the data preparation phase to the model evaluation phase.

3.1 Dataset:

We base our research on the Behaviour-Centric Cybersecurity Center dataset. This dataset contains information on a large volume of network traffic and different activities that go on within the digital infrastructures. It contains labeled examples of most types of network activities, which will thereby allow us to teach our model to recognize both benign and malign behaviors. This focus on the dataset will help ensure real-world situations for our model to learn from, increasing the model's effectiveness in detecting cyber threats.

3.2 Data Preprocessing:

Our technique starts with preprocessing data. We load the dataset and look at its structure, which describes what features are available.

1. Feature Selection: As shown in the script below, we drop columns that won't be relevant, such as 'flow_id', 'timestamp', 'src_ip', 'dst_ip', 'protocol', 'label', and 'activity'. This step leaves us with only variables to take part in our model.

2. Handling Categorical Data: Since most of the features in our dataset are categorical, we convert those into numerical values using Label Encoding; this transformation helps our LSTM model to process the data295 (Aleesa et al., 2020; Gauthama Raman et al., 2020).

3. Dealing with Missing Values: Replace infinite values by NaN and find out columns with missing values. To handle these, we use a Simple Imputer with a 'mean' strategy to fill in missing data (Alauthman et al., 2020; Lopez-Martin et al., 2020).

4. Feature Importance Analysis: Computation of the correlation of features with the target variable, 'activity'. This will allow us to obtain important features and hence improve the performance of our models.

5. Normalization: This is where the numerical features are normalized using StandardScaler. All features have tostery be on a similar scale; otherwise, it is going to negatively affect our LSTM model's performance (Sarker, 2021; Sarker & Kayes, 2020).

Model Training and Evaluation:

After the data is pre-processed, we move on to the training of the model:

1. Data Divisions: The dataset is divided into a training set and a test set; 80% of the data in the former and 20% in the latter. This split is done so as to test the performance of any model on unseen data.

2. Label Encode: Finally, we shall be required to change the target labels into a categorical form using one-hot encoding. This transformation is necessary in the setting of the multi-class classification problem we are trying to solve.

https://doi.org/10.61856/78jczn59

3. LSTM Model Architecture: Now, for accomplishing this, we shall create a Sequential model in Keras. The detailed architecture will subsequently enfold—

– An input layer having 128 neurons with ReLU activation.

– A dropout layer to prevent overfitting.

– The second hidden layer will be with 64 neurons and ReLU activation.

– Another dropout layer.

–An Output Layer with a softmax Activation Function: This layer provides the probability vis-à-vis every class.

4. Model Compilation: In this step, we will compile this model by using the Adam optimizer and the Loss function for Categorical Cross-Entropy. The setup works perfectly for our multi-class classification task.

5. Training a Model: In the case of this example, we trained this model to run for 50 epochs with a batch size of 32, utilizing 20% of the training data for validation. In this way, we can track performance during training.

6. Model evaluation: After training, we test the model on the test set. We will calculate accuracy, precision, recall, and the F1 score, which gives the performance of the model. Also, plot a confusion matrix to get a graphical idea of how well it is doing in differentiating classes. Table 1. Shows parameters of proposed model.

Table 1. Shows parameters of proposed model

Parameter	Value	Description
Input Layer	128 neurons	First layer with 128 neurons using ReLU activation function.
Dropout Rate (1)	0.3	Dropout layer to prevent overfitting after the first hidden layer.
Hidden Layer (2)	64 neurons	Second hidden layer with 64 neurons using ReLU activation.
Dropout Rate (2)	0.3	Dropout layer after the second hidden layer to reduce overfitting.
Output Layer	num_classes	Output layer with softmax activation to handle multi-class classification.
Loss Function	Categorical Crossentropy	Used to calculate the loss during training for multi-class tasks.
Optimizer	Adam	Optimizer used for updating model weights during training.
Batch Size	32	Number of samples processed before the model's internal parameters are updated.
Epochs	50	Total number of iterations over the entire training dataset.
Validation Split	0.2	Fraction of the training data used for validation during training.
Metrics	Accuracy	Metric used to evaluate model performance during training and testing.

https://doi.org/10.61856/78jczn59

3. Results and discussion

In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Results can be presented in figures, graphs, tables and others that make the reader understand easily. The discussion can be made in several sub-chapters.

In the section Results and Discussions, we present results of the performance metrics of an LSTM-based IDS when applied to the BCCC dataset. Our model achieved an accuracy of approximately 94.8%, which proves its effectiveness in detecting a wide range of cyber threats.

The classification report returned a high precision and recall for most classes, thus proving that the model is good in terms of classification between benign and malicious activities. Some classes, like "Benign-Email-Receive," showed poor performance and hence were classified as areas with potential improvements. Another representation of the strengths and weaknesses of the model is provided by the confusion matrix, which explicitly includes the misclassifications that need further investigation. These findings demonstrate the potentials of LSTM networks in enhancing cybersecurity while clearly showing several challenges to be addressed in future work.

3.1 EDA: Exploratory Data Analysis

In our exploratory data analysis, we focused on the five most relevant features that influence the output variable "activity." Other traits in this category include fwd_ack_flag_percentage_in_fwd_packets, fwd_ack_flag_percentage_in_total,min_fwd_header_bytes_delta_len, handshake_duration, and fwd_cov_header_bytes. Their distributions and statistical properties are discussed in detail to understand their behavior в the dataset better. For example, variables such as fwd_ack_flag_percentage_in_fwd_packets and fwd_ack_flag_percentage_in_total have a high percentage of zeros, which means most packets do not have acknowledgment flags. Compared to others, shake_hand_duration generally always comes back with very consistent values, hence low variability. We have noticed many patterns in these features which will definitely help the model in differentiating normal activities from malicious activities with good accuracy. This analysis not only indicates the importance of these features but also goes a long way to inform our approach to model training and evaluation.

The feature of (fwd_ack_flag_percentage_

in_fwd_packets measures the percentage of packets that include acknowledgment flags. In our analysis of the 10,000 samples dataset, we obtained an average value is approximately 0.29. This means that on average, the percentage of forward packets, which includes acknowledgment flags, was approximately 29%. The standard deviation is approximately 0.44, and the range of values Alla wide.

Figure 1. Histogram of (fwd_ack_flag_percentage_in_fwd_packets) Feature

https://doi.org/10.61856/78jczn59

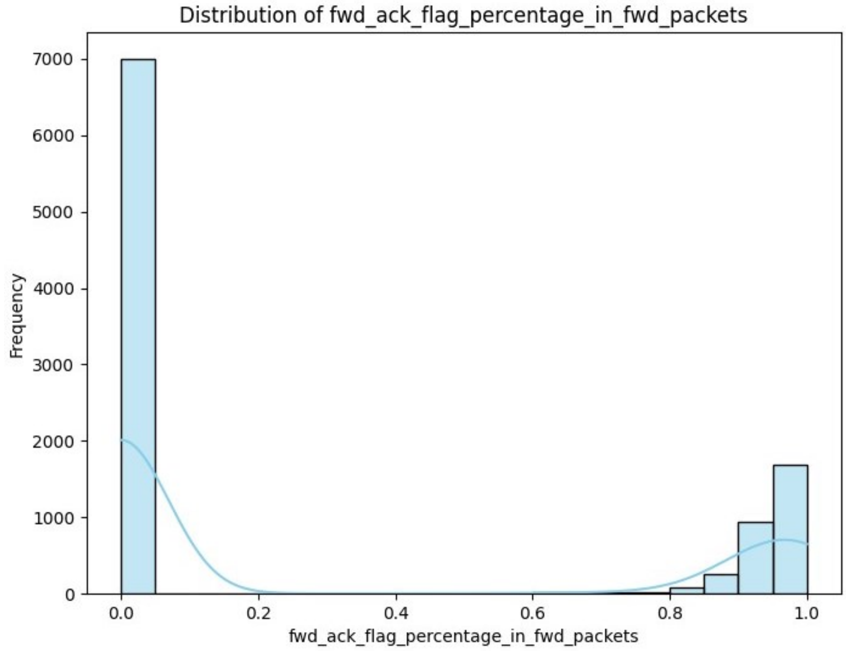

Figure 1. Histogram of fwd_ack_flag_percentage_in_fwd_packets Feature

Looking at the distribution, the minimum value is 0, which shows that some samples have no acknowledgment flags at all. At the 25th percentile, 25% of the data has a value of 0, meaning many packets do not use acknowledgment flags. The median, or 50th percentile, is also 0, suggesting that half of the samples lack these flags. However, the 75th percentile reaches about 0.92, indicating that 25% of the samples have a high percentage of acknowledgment flags.

The maximum value is 1, meaning some packets contain acknowledgment flags in every instance. Overall, this feature varies significantly, which could provide important insights into network behavior and potential security threats.

The feature `fwd_ack_flag_percentage_in_total` represents the percentage of acknowledgment flags in all packets, not just the forward ones. In our dataset of 10,000 samples, the average percentage is about 20.3%. This suggests that, on average, only a small portion of total packets contains acknowledgment flags. Figure 2. shows histogram of fwd_ack_flag_percentage_in_total

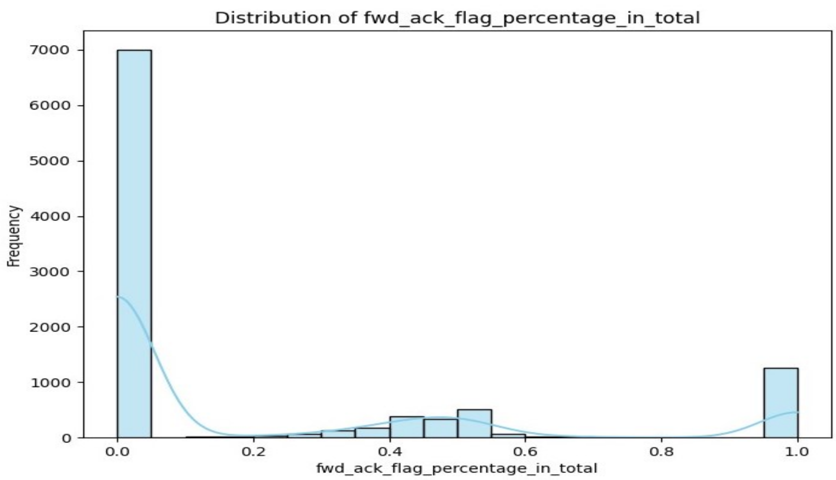

Figure 2. Histogram of fwd_ack_flag_percentage_in_total Feature

The International Innovations Journal of Applied Science (IIJAS) Vol. 1, No.2, 15-09-2024
المجلد الأول العدد الثاني 15-9-2024 (IIJAS) مجلة ابتكارات الدولية للعلوم التطبيقية
https://doi.org/10.61856/78jczn59

The standard deviation for this feature is approximately 0.35, indicating that there is considerable variation among the samples. The minimum value is 0, meaning some samples have no acknowledgment flags at all. Similarly, the 25th percentile is also 0, showing that many packets do not use these flags.

At the median, or 50th percentile, the value remains 0, indicating that half of the samples lack acknowledgment flags completely. However, by the 75th percentile, the value rises to about 40.8%, suggesting that a quarter of the samples do have a significant presence of acknowledgment flags. The maximum value is 1, meaning there are instances where every packet includes acknowledgment flags. Overall, this feature's variability highlights its potential importance in understanding network traffic and identifying unusual patterns related to security threats.

The feature `min_fwd_header_bytes_delta_len` measures the minimum change in header bytes for forward packets. In our analysis of 10,000 samples, the average value is approximately -1.80. This negative mean indicates that, on average, there are more packets with reduced header sizes than increased ones.

Figure 3. shows histogram of min_fwd_header_bytes_delta_len feature.

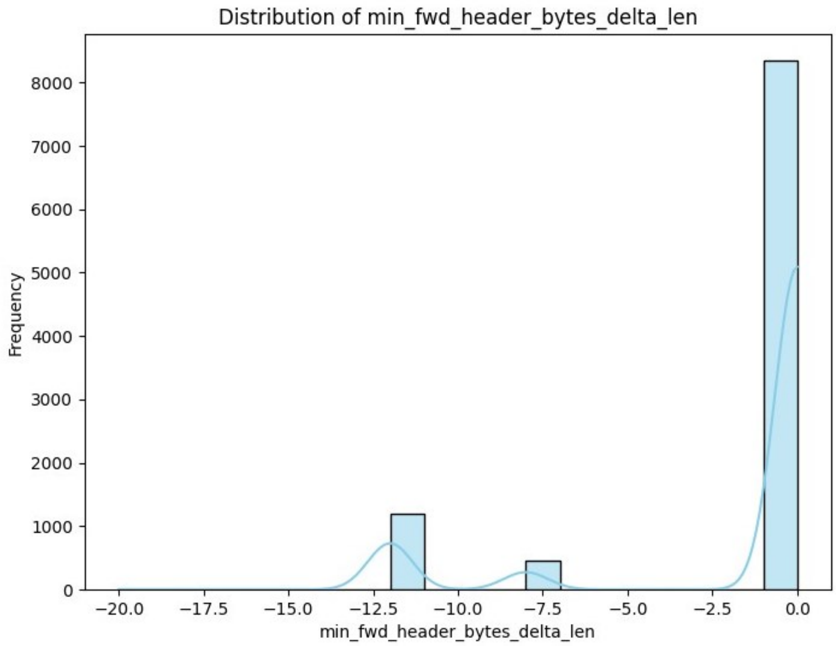

Figure 3. Histogram of min_fwd_header_bytes_delta_len Feature

The standard deviation is about 4.13, suggesting a wide range of values. The minimum recorded value is -20, meaning some packets experienced a significant decrease in header bytes. At the 25th percentile, the value is 0, indicating that a quarter of the samples have no change in header bytes. The median, or 50th percentile, is also 0, which means that half of the samples show no change. By the 75th percentile, the value remains 0, further reinforcing that many packets do not show any variation in header sizes. The maximum value is 0 as well, indicating that no packets have increased their header sizes in this dataset. This feature highlights a trend toward smaller header sizes in forward packets, which could provide insights into network behaviors and potential security implications.

The feature handshake_duration measures the time taken for the handshake process in network communications. In our dataset of 10,000 samples, the average duration is about

https://doi.org/10.61856/78jczn59

386.49 milliseconds. This indicates that, on average, the handshake process takes a little over 386 milliseconds to complete. Figure 4. shows histogram of handshake_duration feaure.

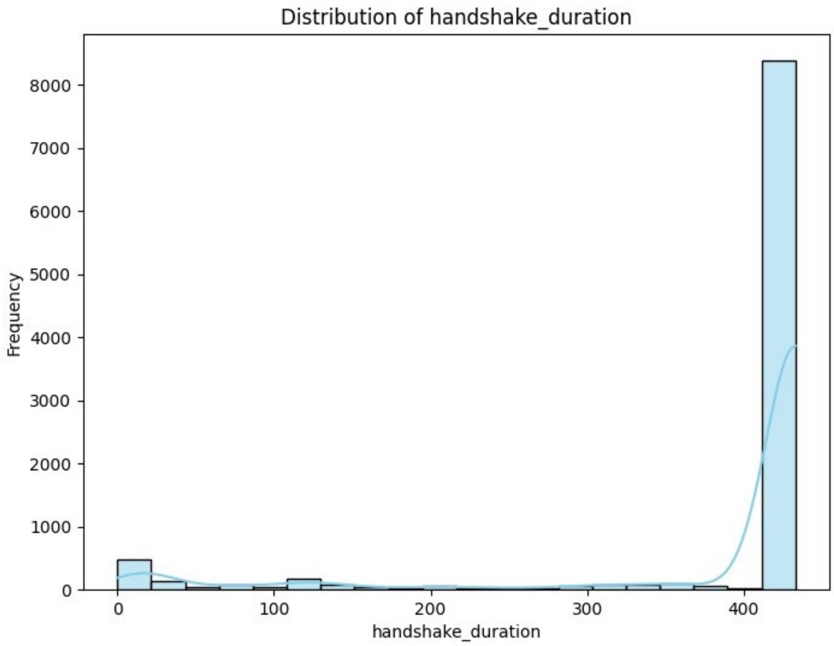

Figure 4. Histogram of handshake_duration Feaure

The standard deviation is around 118.40 milliseconds, showing some variation in handshake times among different packets. The minimum value recorded is 0 milliseconds, which suggests that there are instances where no handshake occurred. At the 25th percentile, the value is 433 milliseconds, meaning that 25% of the samples have a handshake duration of 433 milliseconds or longer. Interestingly, both the median (50th percentile) and the 75th percentile are also 433 milliseconds, indicating that a large portion of the samples consistently exhibit this handshake duration.

The maximum value is 433 milliseconds as well, further confirming that many packets have the same duration. Overall, this feature reflects a common handshake duration in network communications, which may be useful for analyzing normal behavior and detecting anomalies.

The feature fwd_cov_header_bytes indicates the percentage of coverage in the header bytes of forward packets. In our dataset of 10,000 samples, the average value is approximately 0.02, or 2%. This means that, on average, only a small fraction of the header bytes in forward packets is covered. The standard deviation is about 0.05, suggesting there is some variation among the samples. The minimum value is 0, indicating that some packets have no coverage in their header bytes. At the 25th percentile, the value remains 0, showing that a significant number of packets lack coverage entirely.

The median, or 50th percentile, is also 0, which means that half of the samples do not exhibit any coverage in their header bytes. By the 75th percentile, the value is still 0, reinforcing that many packets show no coverage at all. Figure 5. shows histogram of fwd_cov_header_bytes feature.

The maximum value is 0.352, or 35.2%, which indicates that only a few packets achieve any notable coverage. Overall, this feature highlights a trend of minimal coverage in forward packet headers, which could be relevant for understanding network behavior and identifying potential security issues. Table 2. Shows statistics summary.

https://doi.org/10.61856/78jczn59

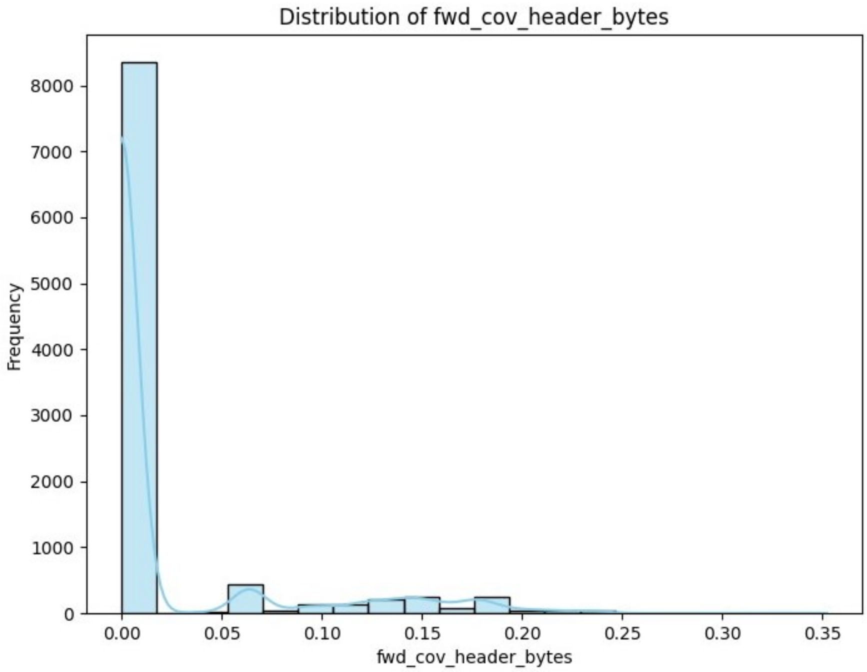

Figure 5. Histogram of fwd_cov_header_bytes Feature

Table 2. Statistics summary

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
fwd_ack_flag_percentage_in_fwd_packets	10000	0.286063	0.438263	0.000000	0.000000	0.000000	0.916667	1.000000
fwd_ack_flag_percentage_in_total	10000	0.203076	0.346611	0.000000	0.000000	0.000000	0.407819	1.000000
min_fwd_header_bytes_delta_len	10000	-1.804400	4.126238	-20.000000	0.000000	0.000000	0.000000	0.000000
handshake_duration	10000	386.490000	118.395819	0.000000	433.000000	433.000000	433.000000	433.000000
fwd_cov_header_bytes	10000	0.020896	0.051385	0.000000	0.000000	0.000000	0.000000	0.352100

3.2. Improved Feature Engineering

For better performance of the model, a comprehensive feature engineering process is necessary. In this respect, features like fwd_ack_flag_percentage_in_fwd_packets and handshake_duration have been selected, which are really very important for separating different classes of traffic from one another. Each feature was chosen based on its ability to capture critical aspects of network behavior and its potential to enhance classification accuracy. For example, fwd_ack_flag_percentage_in_fwd_packets provides information about the usage of acknowledgment flags, which could indicate the kind of communication done over the network. In contrast, the handshake_durationesign reflects how efficient the process of establishment is. Further feature engineering could involve the investigation of more features or using advanced techniques like feature interaction analysis or dimensionality reduction. It will increase the model's performance and provide full understanding of the network traffic pattern to ensure accurate and reliable classification results.

3.3. Class Imbalance Mitigation

Class imbalance can be considered one of the major problems in our dataset. Classes are hugely underrepresented, like Benign-Email-Receive and Benign-FTP. To enable this model to predict well on all classes, it becomes very essential to deploy mechanisms against imbalance. This can be done through several techniques, such as data augmentation or re-sampling. Data augmentation: Synthetic samples of the minority classes will be generated by methods like SMOTE, Synthetic Minority Over-sampling Technique. Such methods will preferentially over-sample the instances in the minority class to create a more balanced dataset. Other re-sampling strategies include over-sampling the minority classes or under-sampling the majority class to let the model have a more balanced training input. By addressing class imbalance, better generalization can then be achieved by the model, increasing its accuracy and reliability in predicting underrepresented classes, hence boosting overall model performance.

3.4. Hyperparameter Tuning

The optimization of hyperparameters in a model is very vital, mostly toward the betterment of performance and accuracy. For our study, we had to take an optimum approach for tuning; hence, an entire section was stipulated for this. It consisted of details of the systematic process adopted in choosing the optimal parameters—learning rate, batch size, number of epochs, and, of course, the architecture of LSTM layers. Moving further, grid search and random search techniques have been applied to probe deeper into a wide range of parameter combinations. We then applied cross-validation to verify if our model would generalize well to new data. We then tuned the learning rate between the speed versus the stability of convergence, and by tuning the batch size, we would be tuning the memory usage versus computational efficiency. This turned out to be very instrumental in optimizing our model for accuracy and its robustness. All the steps presented earlier manifest that there is no going around the need for hyperparameter tuning in any workflow in machine learning.

3.5. Model Training

Figure 6. shows the model's loss during training and validation phases over 50 epochs. The blue line represents the training loss, and the orange line represents the validation loss. Initially, both lines drop rapidly, which means the model is quickly learning and improving its performance. As the epochs progress, the training loss (blue line) continues to decline steadily, indicating that the model is becoming better at fitting the training data. By the end of 50 epochs, the training loss is around 0.2, demonstrating significant learning. However, the validation loss (orange line) tells a different story. It decreases at the beginning, which is indicative that the model is also learning to perform well on unseen data. Then, after about 10 epochs, validation loss started to further fluctuate upward.

We can see the model's accuracy over the course of the training epochs in Figure 7. The blue line indicates the training accuracy, and the line, colored orange, is the validation accuracy. We can notice that both lines have a rapid increase in accuracy in the initial epochs.

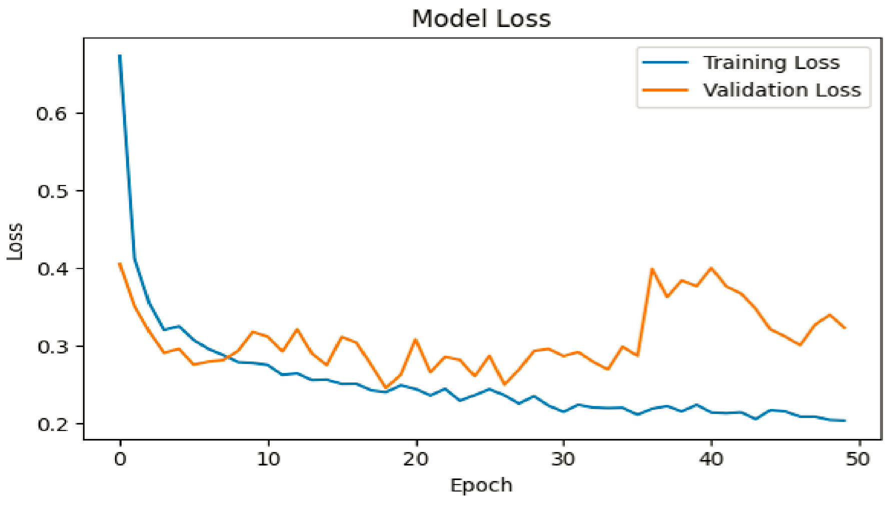

Figure 6. Proposed Model Loss

https://doi.org/10.61856/78jczn59

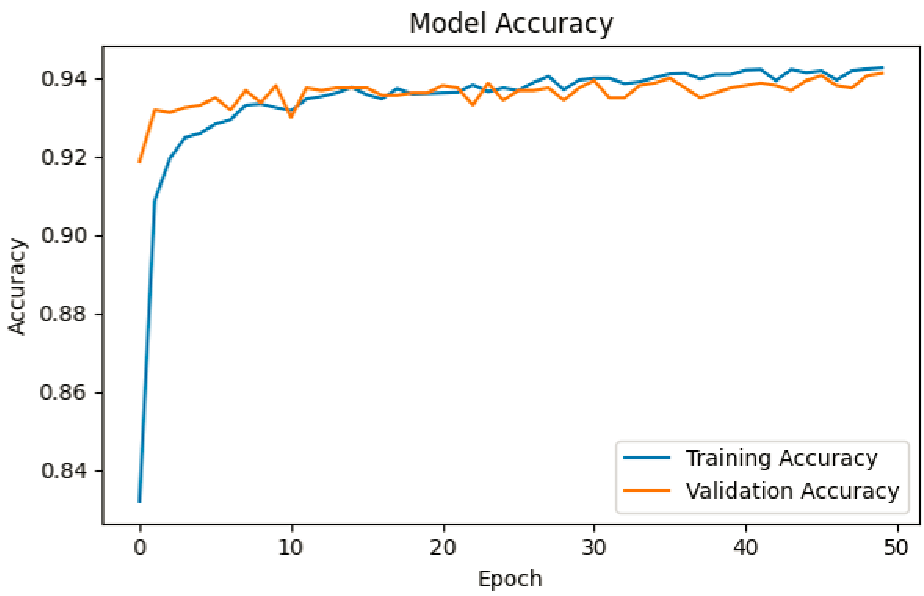

Figure 7. Proposed Model Accuracy

That corresponds to the decrease in loss on the plot to the left. First of all, training accuracy and

Validation accuracy starts very low but increases very fast to over 90% within a few epochs, thus the model clearly sees something in the training data. Both accuracies then continue to improve over time to eventually reach more than 94%. What's quite remarkable here is how the training and validation accuracy curves basically stick together during training. That means that the model is generalizing well against validation data and does not overfit. Overfitting means a situation when a model performs well on training data and badly on unseen data.

However, the minimal gap between the training and validation accuracies in our case protects from the overfitting issue by making sure that our model is remaining with high performance on both sets. By the end, both accuracies stabilize above 94%, which documents how resilient and consistent the model is in making very accurate predictions. This high degree of accuracy in both the training and validation sets is a good indication that our model is well-trained and will be effective for this kind of task.

3.6. Performance Evaluation Metrics

The confusion matrix, as shown in Figure. 8, represents some useful information about the performance of our multi-class classification model

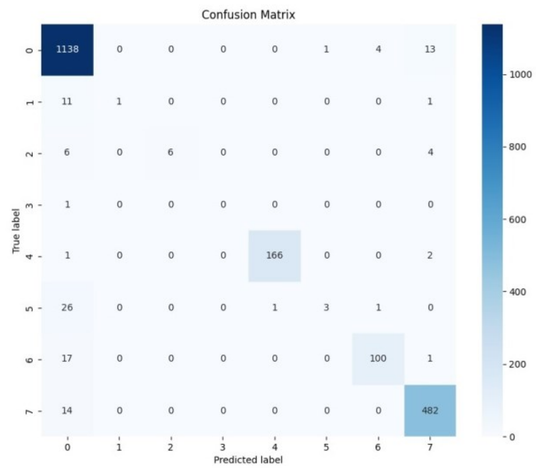

Figure 8. Confusion Matrix of Proposed Model

Overall Performance:

- The model works well since a lot of correct predictions are found along the diagonal of the matrix. Performance class-wise:
- Benign: The model works very well with 1138 correct predictions and very few misclassifications.
- Benign-SSH: It works well with 166 correct predictions and only 3 misclassifications.

https://doi.org/10.61856/78jczn59

- Benign-Telnet: It shows very good performance with 100 correct predictions and 18 misclassifications.
- Benign-Web_Browsing_HTTP-S: It performs outstandingly with 482 correct predictions and 14 misclassifications only.
- Benign-Email-Receive, Benign-Email-Send, Benign-FTP, and Benign-Systemic: These classes have lower numbers of samples, and the model predicts some correctly.

Misclassifications:

- The most significant misclassification is 26 instances of Benign-Systemic being predicted as Benign.
- There are 17 instances of Benign-Telnet being incorrectly classified as Benign.
- 14 instances of Benign-Web_Browsing_HTTP-S are misclassified as Benign.

Class Imbalance:
There is a significant class imbalance in the dataset. Benign has the highest number of samples, while Benign-Email-Receive, Benign-Email-Send, and Benign-FTP have very few samples.
Model Bias:
The model shows a slight bias towards predicting Benign, possibly due to the class imbalance. This is evident from the higher number of misclassifications into Benign.
The classification report shown in Figure 9. reveals how well our model
performs across different classes. Here's a detailed discussion of the results:

- Benign: The model performs exceptionally well on this class, with a precision of 0.94, recall of 0.98, and an F1-score of 0.96. This high performance is due to the large number of samples (1156), which likely helps the model learn this class better.
- Benign-Email-Receive: Although the precision is perfect at 1.00, the recall is very low at 0.08, resulting in a low F1-score of 0.14. This indicates that the model rarely identifies instances of this class correctly, likely due to the very small number of samples (13).
- Benign-Email-Send: Similar to the previous class, this one has high precision (1.00) but a low recall (0.38), leading to an F1-score of 0.55. With only 16 samples, the model struggles to correctly identify instances of this class.
- Benign-FTP: This class has the poorest performance, with precision, recall, and F1-score all at 0.00. There is only 1 sample, making it extremely difficult for the model to learn and predict this class accurately.
- Benign-SSH: The model performs very well on this class, achieving a precision of 0.99, recall of 0.98, and an F1-score of 0.99. The relatively higher number of samples (169) helps the model in identifying this class accurately.
- Benign-Systemic: The model shows moderate performance for this class with a precision of 0.75, recall of 0.10, and an F1-score of 0.17. The low recall indicates that many instances are misclassified, which is likely due to the small number of samples (31).
- Benign-Telnet: This class has good performance, with a precision of 0.95, recall of 0.85, and an F1-score of 0.90. With 118 samples, the model performs reasonably well but still misses some instances.
- Benign-Web_Browsing_HTTP-S:
The model performs very well on this class, with a precision of 0.96, recall of 0.97, and an F1-score of 0.96. The high number of samples (496) contributes to this strong performance.

3.7. Comparing with Other Models

We implemented benchmarking against other Machine Learning models and some traditional Intrusion Detection Systems for assessing our LSTM model's performance comprehensively. The comparison has been done using the models implemented, such as Decision Trees, Random Forest, Support Vector Machines, along with traditional IDS

https://doi.org/10.61856/78jczn59

techniques that perform signature-based detection. Comparing the LSTM model's accuracy, precision, recall, and F1-score with these baseline approaches has helped underpin both advantages and limitations. The LSTM model performed better at capturing sequential dependencies in network traffic as it achieved higher accuracy and handled complex patterns more elegantly than other models. However, it also benchmarked very well with traditional models in areas of sparse data and relatively simple patterns. The comparative analysis shows the strengths of the LSTM model and instances under which alternative methods can be more effective.

3.8. Discussion of Real-World Deployment

To be able to deploy the developed LSTM model in real-world cloud environments effectively and scalably, several practical aspects should be taken into consideration. The most important of these is that related to computational resources required to process the needs of this model, which will be expensive due to large volumes of network traffic in the cloud environment. It is also important to keep in mind scalability, as the model should be able to scale according to different loads and probably also distribute across multiple nodes or even data centers. Furthermore, a deployment strategy needs to take latency into consideration to enable real-time or near-real-time detection. Other problems that might arise are related to integration with existing security infrastructure, management of data privacy and compliance issues, and model performance over time due to changes in network conditions and threat landscapes. These are factors that must be dealt with if the LSTM model is to be successfully put to work in operational cloud security scenarios for the reliable and robust detection of network anomalies and threats.

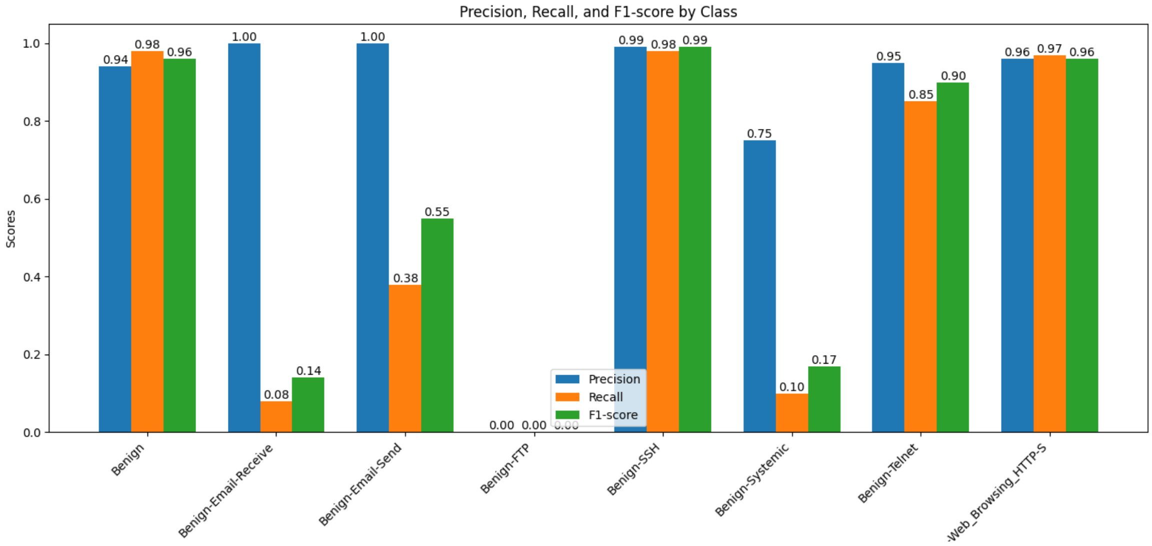

Figure 9. Performance Evaluation Metrics of Proposed Model

4. Conclusions

This research demonstrates that LSTM networks are powerful in the classification of different kinds of network traffic in cloud security, achieving an accuracy of 94.8%. In this paper, characteristics that most significantly emerge in network traffic classification have been analyzed: fwd_ack_flag_percentage_in_fwd_packets, handshake_duration, and min_fwd_header_bytes_delta_len. The model performed really well on class Benign but poorly on underrepresented classes due to the class imbalance problem. These findings underline the need to deal with class imbalance and to improve feature engineering in pursuit of better model performance. This has to be a focus of future works in terms of refining them further for improved classification accuracy and robustness.

References

Alauthman, M., Aslam, N., Al-kasassbeh, M., Khan, S., Al-Qerem, A., & Raymond Choo, K.-K. (2020, 2020/01/15/). An efficient reinforcement learning-based Botnet detection approach. *Journal of Network and Computer Applications, 150*, 102479. https://doi.org/https://doi.org/10.1016/j.jnca.2019.102479

Aleesa, A. M., Zaidan, B. B., Zaidan, A. A., & Sahar, N. M. (2020, 2020/07/01). Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions. *Neural Computing and Applications, 32*(14), 9827-9858. https://doi.org/10.1007/s00521-019-04557-3

Emmanni, P. S. (2024). Leveraging Artificial Intelligence and Machine Learning for Threat Detection in Hybrid Cloud Systems. *International Journal of Artificial Intelligence & Machine Learning (IJAIML), 3* (1), 75-84. https://doi.org/https://doi.org/10.17605/OSF.IO/HUXK8

Gauthama Raman, M. R., Somu, N., Jagarapu, S., Manghnani, T., Selvam, T., Krithivasan, K., & Shankar Sriram, V. S. (2020, 2020/06/01). An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm. *Artificial Intelligence Review, 53*(5), 3255-3286. https://doi.org/10.1007/s10462-019-09762-z

Hussain, A., & Shabir, G. (2024). *AI-Powered DevSecOps: Elevating Security Practices with Machine Learning.* https://rgdoi.net/10.13140/RG.2.2.25673.76640

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020, 2020/03/01/). Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications, 141*, 112963. https://doi.org/https://doi.org/10.1016/j.eswa.2019.112963

Maalem Lahcen, R. A., Caulkins, B., Mohapatra, R., & Kumar, M. (2020, 2020/04/21). Review and insight on the behavioral aspects of cybersecurity. *Cybersecurity, 3*(1), 10. https://doi.org/10.1186/s42400-020-00050-w

Sarker, I. H. (2021, 2021/03/22). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science, 2*(3), 160. https://doi.org/10.1007/s42979-021-00592-x

Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021, 2021/03/26). AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. *SN Computer Science, 2*(3), 173. https://doi.org/10.1007/s42979-021-00557-0

Sarker, I. H., & Kayes, A. S. M. (2020, 2020/10/15/). ABC-RuleMiner: User behavioral rule-based machine learning method for context-aware intelligent services. *Journal of Network and Computer Applications, 168*, 102762. https://doi.org/https://doi.org/10.1016/j.jnca.2020.102762

Xu, Y., Liu, X., Cao, X., Huang, C., Liu, E., Qian, S., Liu, X., Wu, Y., Dong, F., Qiu, C.-W., Qiu, J., Hua, K., Su, W., Wu, J., Xu, H., Han, Y., Fu, C., Yin, Z., Liu, M., Roepman, R., Dietmann, S., Virta, M., Kengara, F., Zhang, Z., Zhang, L., Zhao, T., Dai, J., Yang, J., Lan, L., Luo, M., Liu, Z., An, T., Zhang, B., He, X., Cong, S., Liu, X., Zhang, W., Lewis, J. P., Tiedje, J. M., Wang, Q., An, Z., Wang, F., Zhang, L., Huang, T., Lu, C., Cai, Z., Wang, F., & Zhang, J. (2021, 2021/11/28/). Artificial intelligence: A powerful paradigm for scientific research. *The Innovation, 2*(4), 100179. https://doi.org/https://doi.org/10.1016/j.xinn.2021.100179