Journal of Theoretical and Applied Information Technology
15th December 2024. Vol.102. No. 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# A HYBRID APPROACH COMBINING LONG SHORT-TERM MEMORY AND RANDOM FOREST FEATURE SELECTION FOR NETWORK INTRUSION DETECTION

**[1]HANDRIZAL, [2]FAUZAN NURAHMADI, [3]ALBERTMAN PUTRA BARASA**

[1,2,3]Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. University No. 9-A, Medan 20155, Indonesia

E-mail: handrizal@usu.ac.id

**ABSTRACT**

Network intrusion detection is essential for identifying suspicious network activity. However, with technological advancements and the increasing need for data, data processing becomes a significant challenge in intrusion detection. This research explores a detection system using Long Short-Term Memory (LSTM) combined with Random Forest-based feature selection (RFUTE) to enhance performance. RFUTE reduced data dimensions by selecting 29 features for binary and 30 for multiclass classification from 43 total features. Tested on the NF-UQ-NIDS-v2 dataset, the system showed significant improvements, achieving 99% accuracy in binary classification and 95% in multiclass classification. Additionally, the AUC-ROC curve of the model with RFUTE showed better performance than the model without RFUTE, with an increase in AUC to 0.999 for binary classification and 0.9987 for multiclass classification. These findings demonstrate that the integration of Long Short-Term Memory networks with Random Forest-based feature selection significantly improves the accuracy and predictive performance of network intrusion detection systems, particularly in large-scale and complex environments. This research contributes to the development of a more accurate and efficient model for network intrusion detection by integrating Long Short-Term Memory networks with Random Forest-based feature selection. This model improves the ability to identify network threats quickly and responsively.

**Keywords:** *Network Intrusion Detection, Binary Classification, Multiclass Classification, LSTM, Random Forest Feature Selection (RFUTE), Feature Selection.*

## 1. INTRODUCTION

In the rapidly evolving digital landscape, network security has emerged as a critical concern for individuals, organizations, and nations alike. Cyber-attacks, ranging from data theft to service disruptions, pose significant threats to the integrity, confidentiality, and availability of information systems [9]. The increasing sophistication of these attacks, such as Distributed Denial of Service (DDoS) and advanced malware, demands robust and adaptive security measures to protect network infrastructure [6]. Network Intrusion Detection Systems (NIDS) have become essential in identifying suspicious activities within network traffic that may indicate potential security breaches [16]. The effectiveness of these systems relies on their ability to recognize and classify abnormal patterns or behaviors, thereby allowing timely and targeted responses to detected threats.

In this context, traffic classification is essential to identify the source of traffic from various applications and network services, ensuring quality management, anomaly detection in network security, and recognizing suspicious traffic patterns [1]. As the complexity of network attacks increases, the volume of data generated by network systems also grows significantly. This large and diverse network data presents substantial challenges in intrusion detection, as it requires extensive computational resources for processing and analysis. Efficient data handling techniques are necessary to ensure accurate and timely detection of intrusions. Feature selection is particularly critical in developing effective intrusion detection systems, as not all features in high-dimensional data are essential. Including irrelevant features can lead to model overfitting, making feature selection vital for reducing data dimensionality and enhancing model efficiency.

In light of these challenges and opportunities, this study leverages the Random Forest algorithm for feature selection, which not only provides accurate predictions through multiple decision trees but also effectively handles imbalanced data [15]. Random Forest is used in this research to perform feature selection by ranking and selecting a subset of features that enhance the accuracy and precision of intrusion detection. The Feature Importance Measure (FIM), commonly employed in Random Forest, assesses the significance of each feature within the model. The selected subset of features is then utilized as input for training a Long Short-Term Memory (LSTM) method, which is well-suited for sequential data like network traffic due to its ability to capture temporal relationships within the data [11].

By incorporating LSTM with Random Forest-based feature selection, this study aims to advance the field of intrusion detection by offering a robust approach to mitigating modern cyber threats.

### 1.1 Research Problem

There are not many network intrusion detection systems that can handle large and complex data effectively, especially in terms of detecting attacks with high accuracy. Many existing solutions still face challenges in handling very large data which can affect performance and detection accuracy. This study examines these problems and applies the Long Short-Term Memory (LSTM) method to handle large sequential data, and uses Random Forest Feature selection to reduce data dimensions and improve model efficiency.

## 2. RELATE WORKS

Recent research [10] demonstrates the potential of optimizing Random Forest (RF) algorithms for feature selection. By enhancing randomness to strengthen each tree and reduce inter-tree correlations, the optimized RF model achieved a sensitivity above 0.8 on expanded datasets. However, this study only uses the random forest algorithm, while our study combines the Random Forest algorithm and Long Short-Term Memory Method. Building on these advancements, [3] proposed a Hybrid Sampling & Feature Selection Random Forest (HF_RF) algorithm. HF_RF demonstrated superior performance compared to traditional approaches, particularly in managing imbalanced datasets. However, this study uses the Hybrid Sampling & Feature Selection Random Forest (HF_RF) algorithm, while our study combines the Random Forest algorithm with the Long Short-Term Memory Method. Moreover, studies in deep learning approaches, such as the work by [2], demonstrate the effectiveness of the Long Short-Term Memory (LSTM) method in intrusion detection systems. Their multilayer LSTM model achieved high accuracy rates of 95% and 96% in binary and multiclass classifications, respectively, showcasing its scalability and robustness in network defense. However, traditional intrusion detection systems often suffer from low detection rates and challenges in handling large data, imbalanced classes, and large feature sets, indicating a critical need for improved models and techniques in the field. Therefore, this study proposes a combination of the Random Forest algorithm and the Long Short-Term Memory Method.

## 3. METHODS

### 3.1 Decision Tree

Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It operates by following a series of nested if-else conditions to make predictions, commonly referred to as Classification and Regression Trees (CART). The structure of a Decision Tree begins with a root node, which represents the initial decision point. From there, branches emerge, representing possible decision paths, leading to either internal nodes (representing further decisions) or leaf nodes, which provide final classifications or outcomes [7] [10]. The general structure of a Decision Tree is depicted in Figure 1.

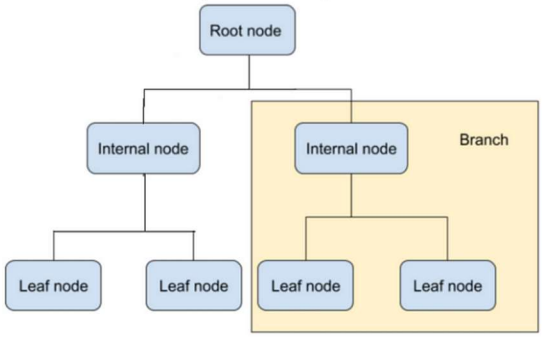


*Figure 1. Structure of Decision Tree* [7]

The process of Decision Tree learning follows a top-down approach known as recursive binary splitting, where the algorithm successively divides the predictor space into two parts. This greedy approach chooses the best split at each step, without looking ahead to optimize the overall tree. The algorithm evaluates variables using statistical criteria to select the optimal split. Two major approaches for selecting splits are based on Entropy and Information Gain, which measure the purity of

the resulting nodes, aiming to create the most homogeneous child nodes.

Entropy measures the level of disorder within a node, with values ranging from 0 (pure) to 1 (completely impure). The entropy at each split helps identify which variable is most effective at dividing the data. Information Gain measures the reduction in entropy from the parent node to the child nodes. It is calculated by subtracting the average entropy of the child nodes from the parent node's entropy. Another commonly used measure is the Gini Index, which focuses on the probability of misclassifying a randomly chosen instance, aiming to minimize misclassification at each split.

Despite its advantages, such as the ability to handle complex, unstructured data (both categorical and numerical) and imbalanced data, Decision Trees have some drawbacks. The most significant challenge is overfitting, where the model becomes too complex and performs poorly on unseen data. Pruning techniques are often applied to reduce tree complexity and prevent overfitting. Furthermore, Decision Trees can be highly sensitive to small changes in the training data, which can lead to significantly different trees. To overcome this, ensemble methods like Random Forest, which aggregates multiple trees, are often employed to improve stability and accuracy.

### 3.2 Random Forest Feature Selection

Random Forest was employed as the method for feature selection due to its effectiveness in identifying the most relevant and informative attributes from a dataset. As an ensemble algorithm that leverages a large number of Decision Trees, Random Forest excels at measuring feature importance through the Feature Importance Measure (FIM), which estimates how frequently a feature is utilized to split the dataset and the extent of impurity reduction produced by that feature [16].

During the feature selection process, features are assessed based on their importance in enhancing detection and classification accuracy. Features with higher information gain are deemed more critical and are retained, while those with lower importance can be eliminated from the dataset. This method not only facilitates the selection of the most relevant feature subset but also helps avoid overfitting by reducing data dimensionality. A detailed illustration of the feature importance in Random Forest can be seen in Figure 2 below.

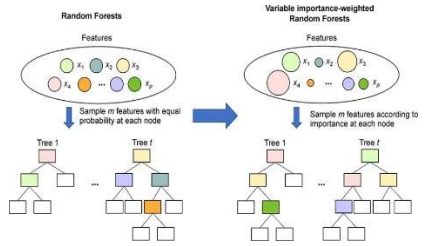


*Figure 2. Feature Importance in Random Forest* [12]

Another advantage of Random Forest is its ability to identify feature interactions and nonlinear relationships between features, which are often missed by other feature selection methods. Several methods have been developed to address the feature selection problem, which can be grouped into three general classes: filter methods, wrapper methods, and embedded methods [8]. This capability provides deeper insights into the factors influencing intrusion detection or classification performance. However, the results from feature selection using Random Forest can vary based on the parameters and configurations employed. Therefore, careful experimentation and validation are necessary to ensure the effectiveness and accuracy of the selected features [15].

Let $A = \{a_1, a_2, \dots, a_d\}$ represent the original available features. The objective of feature selection is to choose a subset of features $A'(|A'| = d', d' \ll d)$ from the original features, meaning $A'$ is a part of $A$. The selected features must have the strongest statistical dependence [5] on the target class $\hat{y}\imath$. In this context, Random Forest performs feature selection by focusing on maximizing the dependence between the selected features $A'$ and the information provided by the target labels $\hat{y}\imath$. This dependence is evaluated using FIM based on information entropy.

Simply put, Random Forest Feature Selection measures the average change in information entropy of each feature by examining how nodes are split across the Decision Trees in the Random Forest. Subsequently, the changes for each feature are ranked from highest to lowest, and the top $n'$ features are chosen to form the target subset $A'$.

The entropy calculation can be expressed as follows:

$$E = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (1)$$

This change is referred to as Information Gain (IG), which measures how much information a feature contributes to predicting the target variable. It is calculated using the following formula:

$$IG = Ent_{parent} - Ent_{chi} \quad (2)$$

Where $Ent_{parent}$ is the entropy of the parent node and $Ent_{child}$ represents the weighted average entropy of the child nodes following the split.

### 3.3 LSTM

The main process in deep learning [3] begins with artificial neural networks posing a series of binary questions, classifying inputs as either true or false. These networks extract numerical values from existing datasets and group them based on the learned information. The final step involves sorting, tagging, and labeling data to ensure accurate and relevant results. This method enables the deep learning model to effectively capture patterns and structure within large datasets.

In particular, Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks (RNN), are highly efficient for processing sequential data, such as text, audio, and time series. LSTM networks have the unique capability to retain information over extended periods, addressing the vanishing gradient problem that often plagues traditional neural networks. This feature allows the LSTM model to capture long-term dependencies within the data, making them ideal for tasks involving time series forecasting, natural language processing, and other sequential data analysis [1].

Combining the strengths of general deep learning methods with the LSTM architecture, researchers can enhance the accuracy and relevance of predictions, especially in domains that rely on sequential data. The binary classification and data grouping mechanisms inherent in the deep learning method, along with LSTM capacity to process time-dependent data, provide a robust framework for analyzing complex datasets. This integration is crucial in applications like network intrusion detection, financial forecasting, and various other fields where pattern recognition across time is essential.

The architecture of an LSTM cell consists of three gates: the forget gate, the input gate, and the output gate. These gates aim to control and protect the cell state while managing the flow of information within the memory cell. With these gates, LSTM can prevent stored information from being overwritten by irrelevant data. Figure 3 below illustrates the architecture of the LSTM cell.

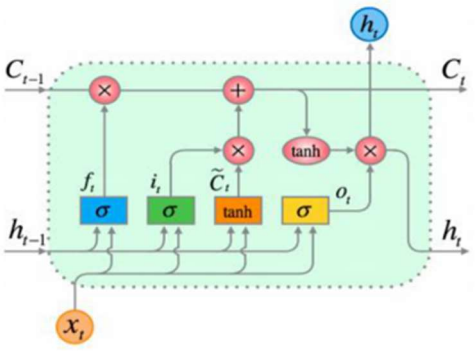


*Figure 3. Architecture of the LSTM Cell* [4]

Figure 3 depicts the architecture of the LSTM cell, where the input layer is represented as $x_t$ and the output is $h_t$ at time $t$. During the model training process and hyperparameter tuning, the LSTM cell takes into account the input cell state $\tilde{C}_t$, the output state from the previous cell $C_{t-1}$, to produce the output state $C_t$. The LSTM cell architecture features three gates: input gate $(i_t)$, forget gate $(f_t)$, and output gate $(o_t)$ (Cui et al., 2018). The formulation of the LSTM cell architecture can be expressed as follows:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{ij}^f x_j^{(t)} + \sum_j W_{ij}^f h_j^{(t-1)}\right) \quad (3)$$

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{ij}^g x_j^{(t)} + \sum_j W_{ij}^g h_j^{(t-1)}\right) \quad (4)$$

$$\tilde{C}_t = \tanh\left(b_i + \sum_j U_{ij}\, x_j^{(t)} + \sum_j W_{ij}\, h_j^{(t-1)}\right) \quad (5)$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{ij}^o x_j^{(t)} + \sum_j W_{ij}^o h_j^{(t-1)}\right) \quad (6)$$

Equations (1) to (4) illustrate the fundamental operations in LSTM. In equation (1) $f_i^{(t)}$ represents the forget gate at time step $t$ for cell $i$. The sigmoid function $(\sigma)$ is employed to adjust weights between 0 and 1. At time step $t$, $x(t)$ is the current input vector, and $h(t)$ is the vector at the hidden layer. Biases $b^f$, $b^g$, $b^o$ are applied to each LSTM cell. Input weights $U^f$, $U^g$, and $U^o$ are added to the current input, while recurrent weights $W^f$, $W^g$, and $W^o$ are added to the hidden layer vector. Equation (3) explains how to update the cell state and timestep in LSTM, while equations (2) and (4) describe the input gate unit $g_i^{(t)}$ and output unit $q_i^{(t)}$ for timestep $t$.

In the context of network intrusion detection, LSTM offers the capability to model complex and dynamic network activity patterns that may change over time. LSTM ability to maintain long-term dependencies makes it an effective choice for building network intrusion detection systems.

### 3.4 Dataset

This research utilizes the NF-UQ-NIDS-v2 dataset, which provides comprehensive information on network attack traffic, categorized to facilitate analysis and application [14]. The NF-UQ-NIDS-v2 dataset includes original NetFlow features as well as modified features. These features include source and destination IP addresses, the protocol used, connection duration, and other key information for detecting network attacks.

The dataset contains a total of 43 feature columns. Of these, 33.12% represent normal (Benign) traffic, while 66.88% are labeled as attack traffic. This large-scale dataset offers a rich foundation for analyzing network behavior and developing intrusion detection models.

### 3.5 Data Preprocessing

Data preprocessing is essential for preparing raw datasets for machine learning models. This process involves separating independent variables, such as IP addresses, protocols, and connection durations, from dependent variables, which represent network activity labels (e.g., benign or attack types). At this stage, the original multiclass classification labels, which consisted of 21 attack categories, were grouped into eight main categories. After this grouping, the process of handling imbalanced data was conducted. For the multiclass classification, manual sampling was applied to ensure a balanced representation of each category. The number of rows for each category was adjusted to achieve a balance between attack categories and normal network activity, allowing for more effective model training.

The dataset is then split into two subsets: 80% for training and 20% for testing. The training set is used to teach the model patterns based on historical data, while the testing set evaluates the model performance on unseen data, ensuring it can be generalized effectively in tasks such as IP-based network behavior detection. An example of the data-splitting process can be seen in Table 1 below.

*Table 1: Separation of IP Address Feature*

| IPV4_SRC_ ADDR | IPV4_ SRC_1 | IPV4_ SRC_2 | IPV4_ SRC_3 | IPV4_ SRC_4 |
|---|---|---|---|---|
| 172.31.69.14 | 172 | 31 | 69 | 14 |

Additionally, categorical variables are encoded using methods such as Label Encoding and One-Hot Encoding to convert non-numeric data into numerical form. Finally, normalization techniques like Standard Scaler are applied to standardize the feature values, ensuring all variables are on a consistent scale by using the following equations:

$$z = \frac{x-\mu}{\sigma} \quad (7)$$

### 3.6 Evaluation Measures

This study involves two types of classification used in network attack detection, namely binary classification and multiclass classification. The evaluation of the model in binary classification is conducted to measure the model performance in distinguishing between two classes: the positive class and the negative class. In contrast, for multiclass classification, the model performance is evaluated based on its ability to differentiate among more than two classes. Key terms for evaluation include True Positive (TP), which refers to cases where the prediction is positive and the actual result is also positive; True Negative (TN), where both the prediction and the actual result are negative; False Positive (FP), where the prediction is positive, but the actual result is negative; and False Negative (FN), where the prediction is negative, but the actual result is positive.

$$Accuracy = \frac{TP+T}{TP+TN+FP+FN} \quad (8)$$

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

$$Recall = \frac{TP}{TP+F} \quad (10)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

A receiver operating characteristic (ROC) curve provides an overview of the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds for performance metrics.

## 4. RESULTS DISCUSSION

This section presents and analyzes the experimental results of applying Random Forest Feature Selection and the LSTM Model for network intrusion detection.

### 4.1 Random Forest Feature Selection

For feature selection, the scikit-learn library was employed, utilizing the Random Forest algorithm to evaluate the importance of each feature in the dataset. The feature selection process involved choosing features with the highest importance scores based on their contribution to classification accuracy. This method was applied for both binary and multiclass classification tasks. The Feature Importance Measure (FIM) scores obtained from this implementation provided a clear understanding of the contribution of each feature in predicting the target labels. The detailed results of feature importance can be seen in Figure 4 for binary

classification and Figure 5 for multiclass classification.

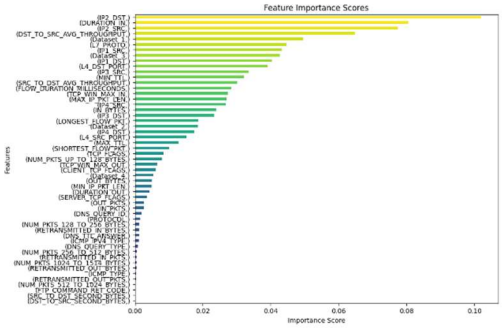


*Figure 4. Random Forest FIM Scores Results in Binary Classification*

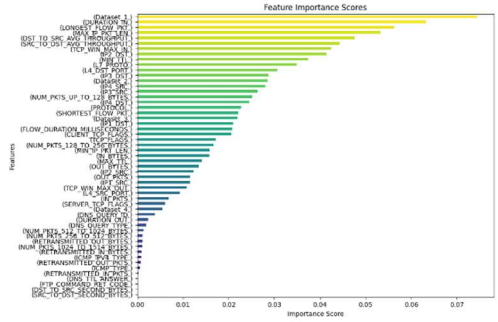


*Figure 5. Random Forest FIM Scores Results in Multiclass Classification*

The next step was to select an optimal subset of features based on the FIM results. To achieve this, the Recursive Feature Elimination with Cross-Validation (RFECV) method was applied using the pre-trained Random Forest model, as implemented in the scikit-learn library. RFECV systematically evaluates the model performance across different feature subsets and recursively removes less important features based on cross-validation results.

The results from the RFECV method demonstrated that for binary classification, 29 features were selected from the initial 43 features, while for multiclass classification, 30 features were chosen from the original set of 43. The selection of these features aimed to optimize model performance by focusing on those that contributed most to classification outcomes. The final selection of features effectively reduced the dimensionality of the data while preserving essential information for classification.

### 4.1 Results of Binary and Multiclass Classification

The model evaluation stage aims to measure the performance of the trained model in both binary and multiclass classification using test data. Then, compares the model performance with and without RFUTE to assess its impact on the model effectiveness.

#### 4.1.1 Binary Classification

In binary classification, the model was evaluated using 400,000 training data points and 100,000 test data points. The evaluation compares the training and validation performance of two models: one with RFUTE (Random Forest-based Feature Selection) and one without RFUTE. This comparison is made to assess the impact of feature selection on model performance.

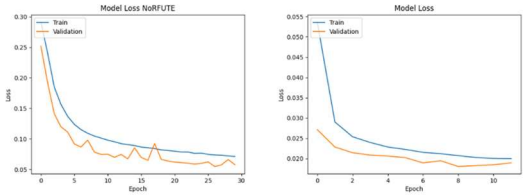


*Figure 6. Comparison of Train and Val Loss in Binary Classification*

Figure 6 illustrates the comparison of training and validation loss between the models trained without and with RFUTE across multiple epochs. For the model without RFUTE, the training loss begins at 0.2917 and decreases to 0.0713 by epoch 30, while the validation loss starts at 0.2519 and drops to 0.0578 in the same period. This indicates that both training and validation loss decrease steadily as the number of epochs increases, although the validation loss remains slightly higher than the training loss. In contrast, the model trained with RFUTE shows a more significant reduction in training loss, starting from 0.0537 at epoch 1 and dropping to 0.0200 by epoch 12. Similarly, the validation loss decreases from 0.0271 to 0.0189 during the same epochs, indicating that RFUTE aids in reducing both training and validation loss more effectively. Furthermore, the gap between training and validation loss is much smaller when using RFUTE, suggesting better generalization.

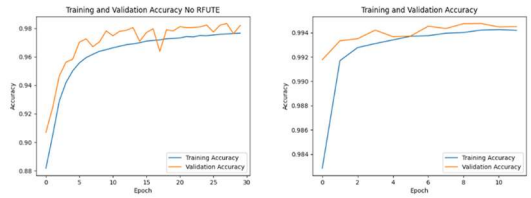


*Figure 7. Comparison of Train and Val Accuracy in Binary Classification*

Figure 7 presents the comparison of training and validation accuracy for the two models. The model without RFUTE shows a training accuracy starting at 0.8818 and improving to 0.9502 by epoch 5, with validation accuracy also increasing from 0.9070 to 0.9583 during the same period. As training progresses, the model training accuracy reaches 0.9766 by epoch 30, while validation accuracy steadily improves from 0.9069 to 0.9819. Conversely, the model trained with RFUTE exhibits significantly higher training accuracy from the first epoch (0.9828), along with a higher validation accuracy of 0.9918. As training continues, the RFUTE-enabled model maintains high training accuracy, reaching 0.9942 by epoch 10, and validation accuracy improves to 0.9947 at the same epoch, remaining stable with slight increases until epoch 12. The classification report further demonstrates that the model with RFUTE achieves superior accuracy, precision, recall, and F1 score, culminating in an overall performance of 99%, compared to the model without RFUTE.

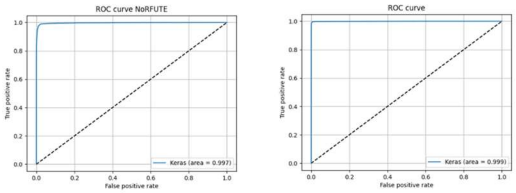


*Figure 8. Comparison of AUC-ROC Curve in Binary Classification*

Figure 8 illustrates the AUC-ROC curve comparing the performance of both models. The model without RFUTE achieves an AUC of 0.997, while the model with RFUTE attains a higher AUC of 0.999, indicating that RFUTE enhances the model's ability to differentiate between classes and improves overall predictive capability.

### 4.1.2 Multiclass Classification

In multiclass classification, the model was evaluated using 303,210 training data points and 75,803 test data points. Similar to binary classification, this evaluation compares the training and validation performance of the two models: one with RFUTE (Random Forest-based Feature Selection) and one without RFUTE.

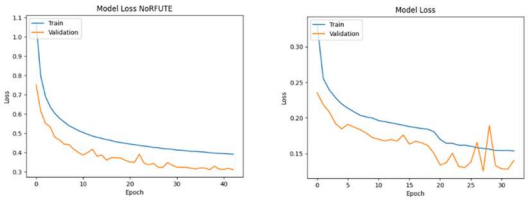


*Figure 9. Comparison of Train and Val Loss in Multiclass Classification*

Figure 9 illustrates the comparison of training and validation loss for both models across various epochs. For the model without RFUTE, the training loss starts at 1.0743 and decreases steadily to 0.3985 by epoch 38. The validation loss also shows a decrease, dropping from 0.7514 at epoch 1 to 0.3108 at epoch 38. This decline indicates that the model gradually improves its ability to minimize errors in predicting both training and validation data. In contrast, the model trained with RFUTE exhibits a significant reduction in training loss, starting from 0.3322 at epoch 1 and falling to 0.1537 by epoch 33. Similarly, the validation loss decreases from 0.2356 at epoch 1 to 0.1403 at epoch 33, demonstrating that the use of RFUTE allows the model to achieve optimal performance more quickly compared to the model without RFUTE.

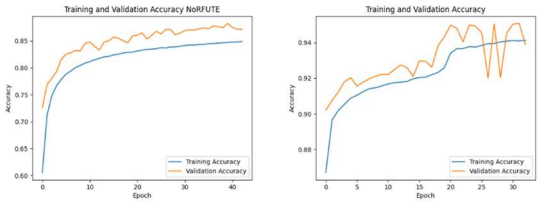


*Figure 10. Comparison of Train and Val Accuracy in Multiclass Classification*

Figure 10 presents the comparison of training and validation accuracy for both models. The model without RFUTE shows a training accuracy beginning at 0.6055, steadily increasing to 0.8460 by epoch 38, while validation accuracy rises from 0.7261 to 0.8765 over the same epochs. In contrast, the model trained with RFUTE displays a significant increase in training accuracy, rising from 0.8670 at epoch 1 to 0.9412 at epoch 33. The validation accuracy also improves from 0.9021 at epoch 1 to 0.9507 at epoch 32. The model reaches its best performance at epoch 28, with the highest validation accuracy before stabilizing in subsequent epochs, indicating that the use of RFUTE enables the model to reach optimal performance faster than the model without RFUTE. The classification report indicates a substantial improvement with RFUTE, showing

higher accuracy, precision, recall, and F1 score, with an average result of 95% compared to 88% for the model without RFUTE.

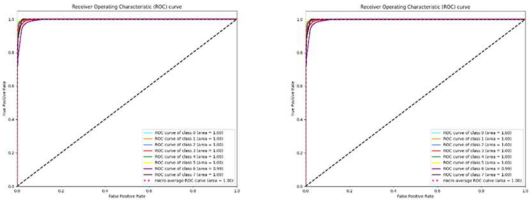

*Figure 11. Comparison of AUC-ROC Curve in Multiclass Classification*

Figure 11 displays the comparison of the AUC-ROC curve for multiclass classification. The results indicate that the application of RFUTE consistently enhances model performance across nearly all classes. Specifically, the micro-average AUC increases from 0.9926 without RFUTE to 0.9987 with RFUTE, reflecting an overall improvement in the model's ability to accurately predict multiple classes. These facts are the key restrictions of this work.

This research contributes to the development of a more accurate and efficient model for network intrusion detection by integrating Long Short-Term Memory networks with Random Forest-based feature selection. This model improves the ability to identify network threats quickly and responsively.

This study uses the NF-UQ-NIDS-v2 dataset from The University of Queensland, Australia which contains network traffic data used to detect network attacks. To check the validity, it is better to try with datasets from other sources. The model is applied with integration with the website interface. Website users can upload a CSV of network data collection to be predicted. To check the validity, it is better to try the Android interface.

## 4. CONCLUSIONS

### 4.1 Conclusions

Based on the results of this study, it show that the methods used by the author have proven effective in improving model performance in predicting various types of attacks more accurately, both in binary classification and multiclass classification to detect network intrusions. This research effectively addresses the challenges of managing large and complex network data for intrusion detection by utilizing Long Short-Term Memory (LSTM) methods and Random Forest Feature Selection (RFUTE) to reduce data dimensionality and enhance model efficiency. RFUTE successfully decreased the dimensionality from 43 features to 29 for binary classification and 30 for multiclass classification. The classification report shows that the model with RFUTE achieved accuracy, precision, recall, and F1-score values as high as 99% in binary classification, while multiclass classification saw average scores improve from 88% to 95%. Additionally, AUC-ROC comparisons indicate that the binary classification model with RFUTE outperforms the model without it, achieving an AUC of 0.999 compared to 0.997, and in multiclass classification, the micro-average AUC increased from 0.9926 to 0.9987. Overall, the methods employed have proven effective in enhancing model performance for accurately predicting various types of attacks in both binary and multiclass classifications for network intrusion detection.

### 4.1 Future Research

For future works, in improving practical applications, the results of this study can be integrated with existing network security systems, such as firewalls or signature-based threat detection systems. This integration will provide a more comprehensive approach to dealing with various network security threats.

It is expected that this study can be further tested in a real network environment, not only on a simulation dataset. Testing on a real network will provide an evaluation of the model's performance in real conditions and provide valuable feedback on the strengths and weaknesses of the developed system.

To facilitate the use of this system, the development of a user-friendly interface for network administrators or network security analysts is also recommended. This interface can provide a real-time display of the prediction model results with detailed visualizations related to the type of attack, risk level, and recommendations for mitigation actions.

## REFRENCES:

[1] Ahmad, R., Alsmadi, I., Alhamdani, W., & Tawalbeh, L. (2022). A Deep Learning Ensemble Approach to Detecting Unknown Network Attacks. *Journal of Information Security and Applications*, *67*, 103196. https://doi.org/10.1016/j.jisa.2022.103196

[2] Bashar, G. M. H., Kashem, M. A., & Paul, L. C. (2022). Intrusion Detection for Cyber-Physical Security System Using Long Short-Term Memory Model. *Scientific Programming*, *2022*. https://doi.org/10.1155/2022/6172362

[3] Cui, H., Xu, H., & Li, J. (2023). Optimization of Random Forest Algorithm Based on Mixed Sampling Additional Feature Selection. 2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE), 461-467. https://doi.org/10.1109/ICCECE58074.2023.10135433

[4] Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*. 1–11. https://doi.org/10.48550/arXiv.1801.02143

[5] Hamla, H., & Ghanem, K. (2021). Comparative Study of Embedded Feature Selection Methods on Microarray Data. Artificial Intelligence Applications and Innovations, 69-77. https://doi.org/10.1007/978-3-030-79150-6_6

[6] Intelligence, M. T. (2020). Microsoft Digital Defense Report. *Network Security*, *2020*(10), 4–4. https://doi.org/10.1016/s1353-4858(20)30114-8

[7] Krueger, E., Bongale, S., & Franklin, D. (2021). Learn how decision trees are grown. Towards Data Science. Accessed on April 30, 2024, from https://towardsdatascience.com/learn-how-decision-trees-are-grown-22bc3d22fb51

[8] Layeb, A. (2023). Novel Feature Selection Algorithms Based on Crowding Distance and Pearson Correlation Coefficient. *International Journal of Intelligent Systems and Applications*, *15*(2), 37–42. https://doi.org/10.5815/ijisa.2023.02.04

[9] Lewis, J. A. (2022). A Shared Responsibility: Public-Private Cooperation for Cybersecurity. Accessed on April 2, 2024, from https://www.csis.org/james-lewis-publications

[10] Li, W. (2022). Optimization and Application of Random Forest Algorithm for Applied Mathematics Specialty. *Security and Communication Networks*, *2022*. https://doi.org/10.1155/2022/1131994

[11] Lindemann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, *99*(July 2020), 650–655. https://doi.org/10.1016/j.procir.2021.03.088

[12] Liu, Y., & Zhao, H. (2017). *Variable importance-weighted Random Forests*. *5*(4), 338–351. https://doi.org/10.1007/s40484-017-0121-6

[13] Roberts, D. A., Yaida, S., & Hanin, B. (2022). The Principles of Deep Learning Theory. *The Principles of Deep Learning Theory*. https://doi.org/10.1017/9781009023405

[14] Sarhan, M., Layeghy, S., & Portmann, M. (2022). *Towards a Standard Feature Set for Network Intrusion Detection System Datasets*. 357–370. https://doi.org/https://doi.org/10.1007/s11036-021-01843-0

[15] Sun, X., & Chai, J. (2023). Random forest feature selection for partial label learning. *Neurocomputing*, *561*(June), 126870. https://doi.org/10.1016/j.neucom.2023.126870

[16] Valdovinos, I. A., Pérez-Díaz, J. A., Choo, K. K. R., & Botero, J. F. (2021). Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges, and future directions. *Journal of Network and Computer Applications*, *187*(May), 103093. https://doi.org/10.1016/j.jnca.2021.103093