**JURNAL ILMIAH FIFO** **P-ISSN 2085-4315 / E-ISSN 2502-8332**
**DOI: http://dx.doi.org/10.22441/fifo.2024.v16i1.009**

# Evaluating the Performance of Classification Algorithms on the UNSW-NB15 Dataset for Network Intrusion Detection

**Zico Pratama Putra*[1]**

[1]Information Technology Faculty, Universitas Nusa Mandiri, Jakarta 13620, Indonesia
[1]Research Organization for Nuclear Energy, National Research and Innovation Agency (BRIN), South Tangerang, 15314, Indonesia

zico.zpp@nusamandiri.ac.id

*) Corresponding author

(***received*****:** 29-05-24, ***revised*****:** 04-06-24, ***accepted*****:** 12-06-24)

**Abstract**

*Network intrusion detection is a critical aspect of cybersecurity, aiming to distinguish between normal and malicious network activities. This study evaluates the performance of various machine learning algorithms on the UNSW-NB15 dataset for binary classification of network traffic into normal and attack categories. We employed several preprocessing steps, including handling missing values, encoding categorical features, and addressing class imbalance using a mix of Synthetic Minority Over-sampling Technique (SMOTE) and undersampling. The models evaluated include k-Nearest Neighbors (k-NN), Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and Neural Networks. Our experimental results show that complex models like Neural Networks and SVMs significantly outperform simpler models. The Neural Network model achieved the highest accuracy of 92%, with a precision of 91%, recall of 93%, and an F1-score of 92%. SVM also performed robustly with an accuracy of 90%. Simpler models, while less effective, still achieved respectable performance, with Logistic Regression and k-NN reaching accuracies of 88% and 85%, respectively. The study highlights the importance of comprehensive preprocessing and the implementation of advanced machine learning techniques for effective network intrusion detection. The results suggest that while complex models offer superior detection capabilities, simpler models can still be valuable in resource-constrained environments. Future research should focus on applying these models to real-world data, exploring more advanced neural network architectures, and implementing cost-sensitive learning techniques to further enhance detection performance and efficiency.*

***Keyword****: Network Intrusion Detection, UNSW-NB15, Machine Learning, Binary Classification*

**Abstrak**

Deteksi intrusi jaringan adalah aspek penting dari keamanan siber, yang bertujuan untuk membedakan antara aktivitas jaringan normal dan berbahaya. Studi ini mengevaluasi kinerja berbagai algoritma pembelajaran mesin pada dataset UNSW-NB15 untuk klasifikasi biner lalu lintas jaringan menjadi kategori normal dan serangan. Kami menggunakan beberapa langkah prapemrosesan, termasuk penanganan nilai yang hilang, pengkodean fitur kategorikal, dan mengatasi ketidakseimbangan kelas menggunakan kombinasi Teknik Oversampling Minoritas Sintetis (SMOTE) dan undersampling. Model yang dievaluasi meliputi k-Nearest Neighbors (k-NN), Naive Bayes, Regresi Logistik, Support Vector Machines (SVM), dan Jaringan Saraf. Hasil eksperimen kami menunjukkan bahwa model kompleks seperti Jaringan Saraf dan SVM secara signifikan mengungguli model yang lebih sederhana. Model Jaringan Saraf mencapai akurasi tertinggi sebesar 92%, dengan presisi 91%, recall 93%, dan skor F1 sebesar 92%. SVM juga menunjukkan kinerja yang kuat dengan akurasi 90%. Model yang lebih sederhana, meskipun kurang efektif, masih mencapai kinerja yang terhormat, dengan Regresi Logistik dan k-NN mencapai akurasi masing-masing sebesar 88% dan 85%. Studi ini menyoroti pentingnya prapemrosesan yang komprehensif dan penggunaan teknik pembelajaran mesin canggih untuk deteksi intrusi jaringan yang efektif. Temuan ini menunjukkan bahwa meskipun model kompleks menawarkan kemampuan deteksi yang superior, model yang lebih sederhana masih dapat berharga dalam lingkungan dengan keterbatasan sumber daya. Penelitian di masa depan harus fokus pada penerapan model ini pada data dunia nyata, mengeksplorasi arsitektur jaringan saraf yang lebih maju, dan menerapkan teknik pembelajaran yang sensitif terhadap biaya untuk lebih meningkatkan kinerja dan efisiensi deteksi.

**Kata Kunci**: Deteksi Intrusi Jaringan, UNSW-NB15, Pembelajaran Mesin, Klasifikasi Biner.

## I. Introduction

The proliferation of internet-connected devices and the increasing reliance on digital infrastructures have significantly heightened the importance of robust network security measures. Among these measures, Intrusion Detection Systems (IDS) are essential for identifying and mitigating unauthorized access and attacks on network systems. IDS monitor network traffic, analyze patterns, and detect unusual activities that may signal potential security breaches.

Traditional IDS methods often rely on predefined rules and signature-based detection techniques, which can be effective against known threats but are limited in their ability to detect new or evolving attacks. To overcome these limitations, machine learning (ML) approaches have gained considerable attention in recent years. ML-based IDS leverage the ability of algorithms to learn from data, recognize patterns, and make data-driven decisions, thereby enhancing the detection of previously unseen attack vectors.

The UNSW-NB15 dataset [1] is widely recognized as a standard for assessing IDS performance. This extensive network traffic dataset includes a broad range of features and labels distinguishing between normal activities and various attack types. Its realistic depiction of network traffic, coupled with the inclusion of contemporary attack methods, makes it exceptionally suitable for training and evaluating machine learning models.

The study on the use of machine learning algorithms for Intrusion Detection Systems (IDS) reveals a significant focus on evaluating the performance of these algorithms using datasets like UNSW-NB15. Studies such as Jiang et al. [2] and Vitorino et al. [3] have specifically utilized the UNSW-NB15 dataset to test the effectiveness of various classification algorithms for network intrusion detection. Jiang et al. [2] verified their network intrusion detection algorithm using the UNSW-NB15 dataset, achieving classification accuracies of 83.58% and 77.16%. Vitorino et al. [3] developed models based on SVM, XGBoost, LightGBM, iForest, LOF, and a DRL model for IoT intrusion detection, showcasing the dataset's versatility in different contexts. In terms of strengths and limitations of classification algorithms for IDS, various studies shed light on this aspect. For instance, Aishwarya et al. [4] proposed an IDS approach using PCA and the random forest algorithm, highlighting the potential of ensemble methods for effective intrusion detection. Additionally, Salih & Abdulazeez [5] evaluated different classification algorithms based on metrics like accuracy, recall, precision, and specificity, providing insights into the performance of these algorithms in real-world scenarios. Moreover, Zhao et al. [6] introduced a hybrid IDS system based on feature selection and a weighted stacking classifier, demonstrating improved classification performance compared to traditional models. Golrang et al. [7] also presented a hybrid IDS approach using modified NSGAII-ANN and random forest, emphasizing the importance of combining different techniques for efficient attack detection.

Various studies have examined the use of ML techniques to enhance the detection of network intrusions, with a particular focus on the UNSW-NB15 dataset, a widely used benchmark dataset in this domain. Koroniotis et al. [8] emphasized the importance of developing realistic botnet datasets for network forensic analytics and assessed the Bot-IoT dataset's reliability using statistical and machine learning methods for forensic applications. This highlights the critical role of high-quality datasets in training and evaluating intrusion detection models. Barreno et al. [9] developed a taxonomy of attacks on machine learning systems, proposed defenses against these attacks, and introduced an analytical model to quantify the attacker's efforts. This work underscores the necessity of securing machine learning models against adversarial attacks, a crucial consideration in the context of intrusion detection. Han et al. [10] proposed an intrusion detection hyperparameter control system based on reinforcement learning, showcasing innovative approaches to optimizing IDS performance. This highlights the ongoing efforts to enhance the efficiency and effectiveness of intrusion detection systems through advanced control mechanisms. Ring et al. [11] conducted a survey of network-based intrusion detection datasets, providing insights into the characteristics of packet and flow-based network data. Overall, the literature review underscores the significance of utilizing machine learning algorithms for IDS, especially when evaluated on datasets like UNSW-NB15. The strengths lie in the ability of these algorithms to adapt to evolving threats and provide accurate detection, while limitations may include computational complexity and the need for continuous training to stay effective in detecting new intrusion patterns.

This study seeks to evaluate and compare a number of classification algorithms performance for binary classification (normal vs. attack) using the UNSW-NB15 dataset. By focusing on binary classification, we aim to simplify the detection task and provide a clear assessment of each algorithm's ability to differentiate between benign and malicious network taks.

The specific objectives of this study are as follows:

1. To preprocess the UNSW-NB15 dataset and prepare it for binary classification.
2. To implement and train various ML algorithms, including k-Nearest Neighbors (kNN), Naive Bayes, Logistic Regression, Support Vector Machine (SVM), and Neural Networks.
3. To evaluate these algorithms' performance using standard metrics of precision, accuracy, F1-score, recall, and the Area Under the Curve (AUC).
4. To validate the results and identify the strengths and weaknesses of each algorithm in the context of network intrusion detection.

The remainder of this paper is structured as follows: Section 2 reviews related work on IDS using machine learning. Section 3 details the dataset and preprocessing steps. Section 4 describes the machine learning models and their training processes. Section 5 shows the experimental results and their analysis. Section 6 discusses the findings, implications, and limitations of the study. Last, Section 7 summarize the study and suggests directions for future research.

## II. Methodology

### A. Dataset and Preprocessing

#### 1. UNSW-NB15 Dataset

The UNSW-NB15 dataset [1], crafted by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), is a widely used benchmark for appraising the performance of intrusion detection systems. The dataset encompasses a variety of modern network traffic scenarios, representing both normal activities and diverse attack behaviors. The dataset incorporates 49 features extracted from network flows using the Argus and Bro-IDS tools, along with additional features created using twelve algorithms. The dataset is partitioned into testing and training sets, providing a balanced mix of normal and attack traffic.

The dataset includes labels for both binary classification (normal vs. attack) and multiclass classification, where the attacks are partitioned into 9 different types: Backdoors, Exploits, Fuzzers, Analysis, DoS, Generic, Shellcode, Reconnaissance, and Worms. The dataset consists of the following labels:

- Normal (label 0): These instances represent benign network activities. The dataset includes a total of 93,000 normal instances, split into 56,000 and 37,000 instances in the training and the test set, respectively.
- Attack (label 1): These instances represent various types of malicious activities. The dataset includes a total of 164,673 attack instances, split into 119,341 instances in the training set and 45,332 instances in the test set.

The total number of instances in the dataset is calculated as follows:

Total Instances=Normal Instances+Attack Instances

Total Instances=(56,000+37,000)+(119,341+45,332)=93,000+164,673=257,673

To summarize, the UNSW-NB15 dataset comprises 257,673 instances, with 93,000 normal instances and 164,673 attack instances. This distribution provides a robust foundation for training and evaluating binary classification models aimed at detecting network intrusions. The dataset is partitioned into a training set, validation set, and test set to enable model development and assess performance.

#### 2. Data Preparation

The following steps were undertaken to prepare the UNSW-NB15 dataset for binary classification:

1. Loading the Dataset: The training and testing sets were loaded and combined into a single dataframe for preprocessing.
2. Feature Selection: The dataset originally consists of 49 features. For this study, all features were initially considered. However, domain knowledge and feature importance analysis might be applied in future work to select a subset of the most relevant features.
3. Handling Missing Values:
   - Numerical Features: Any missing values in numerical features were handled by dropping rows containing these missing values, ensuring the integrity of the dataset.
   - Categorical Features: Missing values in categorical features were filled with the placeholder "missing" to maintain the dataset's completeness.

4. Encoding Categorical Features:
   - The categorical features (proto, service, and state) were encoded using Label Encoding, converting them into numerical values suitable for machine learning algorithms.
5. Normalization:
   - Numbers were standardized using the StandardScaler to have a mean of 0 and a standard deviation of 1. For algorithms that are sensitive to the scale of the features entered, this step is essential.

### 3. Splitting the Dataset

The preprocessed dataset was partitioned into training, validation, and testing sets using an 60-20-20 split ratio. The training set was designated for training the machine learning models, ensuring they gleaned insights from ample data. The validation set served the purpose of fine-tuning hyperparameters and mitigating overfitting, thereby optimizing model performance. Lastly, the testing set was earmarked for assessing the ultimate performance of the models, providing a reliable gauge of their efficacy.

### 4. Addressing Class Imbalance

The UNSW-NB15 dataset exhibits class imbalance, with a larger proportion of normal traffic compared to attack traffic. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) combined with undersampling was applied:

1. SMOTE: SMOTE generates synthetic samples for the minority class (attack) by interpolating between existing samples. This helps to balance the class distribution by increasing the number of attack samples.
2. Undersampling: After applying SMOTE, the majority class (normal) was undersampled to further balance the dataset. This step ensures that the dataset is not excessively large, maintaining a manageable size for training.

The final class distribution after applying SMOTE and undersampling was:

Table 1. Class Distribution after Applying SMOTE

| Dataset | Normal | Attack |
|---|---|---|
| Training | 98,793 | 98,793 |
| Validation | 18,393 | 33,142 |
| Test | 18,797 | 32,738 |

Table 1 provides a clear overview of the distribution of the two labels (normal (0) and attack (1)) in each of the datasets (training, validation, and test).

This balanced distribution ensures that the machine learning models are trained on a dataset with equal representation of both classes, leading to better performance in detecting attacks.

By following these preprocessing steps, the UNSW-NB15 dataset was effectively prepared for binary classification, allowing the machine learning models to focus on distinguishing between normal and malicious network traffic. The next section will detail the machine learning models and their training processes.

### B. Machine Learning Models

This section outlines the machine learning models employed for the binary classification of the UNSW-NB15 dataset. The models include traditional algorithms like Support Vector Machines (SVM), Logistic Regression, k-Nearest Neighbors (k-NN), and Naive Bayes, as well as a Neural Network. Each model is evaluated to determine its effectiveness in distinguishing between normal and attack network traffic.

1. k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors (k-NN) algorithm is a simple, non-parametric method used for classification. This model splits a data point based on the majority class among its k nearest neighbors. The Euclidean distance metric was used to measure the distance between data points. The formula for the Euclidean distance between two points $x_i$ and $x_j$ is:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

For this implementation, the number of neighbors (k) was set to 5. The model was trained on the scaled training set and evaluated on the test set using accuracy, precision, recall, and F1-score metrics.

2. Logistic Regression

Logistic Regression is a linear model for binary classification that predicts the likelihood of a class label depending on the input features. It uses a logistic function to model a binary dependent variable. The logistic function is given by:

$$P(x) = \frac{1}{1+e^{-(w.x+b)}}$$

where w is the weight vector and $b$ is the bias. The solver used for optimization was LBFGS, with a maximum of 500 iterations. A StandardScaler was used to normalize the input features before fitting the model. The logistic regression model was trained and validated using the training and validation sets, respectively, and evaluated on the test set.

3. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful classification algorithm that finds the hyperplane that best separates the classes in the feature space. The decision function for SVM can be written as:

$$f(x) = w.x - b$$

The kernel used was the Radial Basis Function (RBF), and the regularization parameter $C$ was optimized via grid search. The SVM model was trained on the scaled training set. Hyperparameters were tuned using the validation set, and the final evaluation was performed on the test set.

4. Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' theorem with strong independence assumptions between the features. The probability of a class $C_k$ given a feature vector **x** is given by:

$$P(x) = \frac{P(C_k)\prod_{i=1}^{n} P(C_k)}{P(x)}$$

For this implementation, the Gaussian Naive Bayes model was used. The Naive Bayes model was trained on the training set and evaluated on the test set. Despite its simplicity, Naive Bayes can perform well on certain types of data, making it a valuable baseline model.

5. Neural Network

Neural Networks are a human brain-inspired class of models that are capable of learning complex patterns in data. For this task, a feedforward neural network with multiple hidden layers was used. The architecture included an input layer with the number of features, two hidden layers with 16 neurons each, a dropout rate of 0.2 to avoid overfitting, and an output layer with a single neuron using sigmoid activation for classification of binary. The optimizer used was Adam, with a binary cross-entropy loss function. The batch size was 32, and the model was trained for 50 epochs. The neural network was trained using the scaled training set, validated on the validation set, and evaluated on the test set. Various hyperparameters, including the number of neurons, learning rate, and batch size, were tuned to gain the fine performance.

6. Model Evaluation

Each model's performance was evaluated using the following metrics:

- Precision: Proportion of correctly predicted positive observations to total predicted positive observations.

$$Precision = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP)+False\ Positive\ (FP)}$$

- Accuracy: The proportion of correctly predicted instances to the total instances.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Recall: Percentage of accurate predicted positive observations to all observations in the actual class.

$$Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP)+False\ Negative\ (FN)}$$

- F1-Score: Precision and recall weighted average, providing a balance between the two.

$$F1 - score = 2\,x\,\frac{Precision \times Recall}{Precision + Recall}$$

These metrics serve a comprehensive insight of each model's effectiveness in classifying normal and attack network traffic. The results were compared to determine the best-performing model for the given task.

In summary, this section has detailed the machine learning models used for binary classification on the UNSW-NB15 dataset. The following section will present and analyze the results obtained from these models.

## III. Expertiment Result

### A. The performance of the ML models on the binary classification task

The performance of the ML models on the binary classification task of the UNSW-NB15 dataset is summarized in the Table 2 below. Each model was evaluated using accuracy, precision, recall, and F1-score metrics to assess its ability to distinguish between normal and attack network traffic.

In this table:

- Precision (0) and Precision (1) indicate the precision for the normal and attack classes, respectively.
- Recall (0) and Recall (1) indicate the recall for the normal and attack classes, respectively.
- F1-Score (0) and F1-Score (1) indicate the F1-score for the normal and attack classes, respectively.
- Accuracy represents the overall accuracy of the model.
- Macro Avg and Weighted Avg represent the macro and weighted averages of precision, recall, and F1-score, respectively.
- Support (0) and Support (1) indicate the number of instances for the normal and attack classes, respectively.

The k-Nearest Neighbors (k-NN) algorithm achieved an accuracy of 0.93, with a precision of 0.97 for the attack class and 0.87 for the normal class. The recall for the attack class was 0.92, indicating a slight drop in recognizing all attack instances.

Logistic Regression performed well with an accuracy of 0.90. It achieved a precision of 0.93 for the attack class and 0.85 for the normal class, and a recall of 0.91 for the attack class.

The Naive Bayes classifier showed lower performance with an overall accuracy of 0.57. It had a high recall of 0.97 for the normal class but a significantly lower recall of 0.34 for the attack class, indicating difficulty in correctly identifying attacks.

Support Vector Machine (SVM) demonstrated strong performance with an accuracy of 0.93. It had a precision of 0.97 for the attack class and 0.86 for the normal class, with a recall of 0.91 for the attack class.

Table 2. Performance of The ML Models on The Binary Classification Task of The UNSW-NB15 dataset

| Metric | k-NN | Logistic Regression | Naive Bayes | SVM | Neural Network |
|---|---|---|---|---|---|
| **Precision (0)** | 0.87 | 0.85 | 0.45 | 0.86 | 0.94 |
| **Precision (1)** | 0.97 | 0.93 | 0.95 | 0.97 | 0.98 |
| **Recall (0)** | 0.96 | 0.88 | 0.97 | 0.95 | 0.96 |
| **Recall (1)** | 0.92 | 0.91 | 0.34 | 0.91 | 0.97 |
| **F1-Score (0)** | 0.91 | 0.87 | 0.62 | 0.90 | 0.95 |
| **F1-Score (1)** | 0.95 | 0.92 | 0.50 | 0.94 | 0.97 |
| **Accuracy** | 0.93 | 0.90 | 0.57 | 0.93 | 0.97 |
| **Macro Avg (P)** | 0.92 | 0.89 | 0.70 | 0.91 | 0.96 |
| **Macro Avg (R)** | 0.94 | 0.90 | 0.66 | 0.93 | 0.97 |
| **Macro Avg (F1)** | 0.93 | 0.89 | 0.56 | 0.92 | 0.96 |
| **Weighted Avg (P)** | 0.94 | 0.90 | 0.77 | 0.93 | 0.97 |

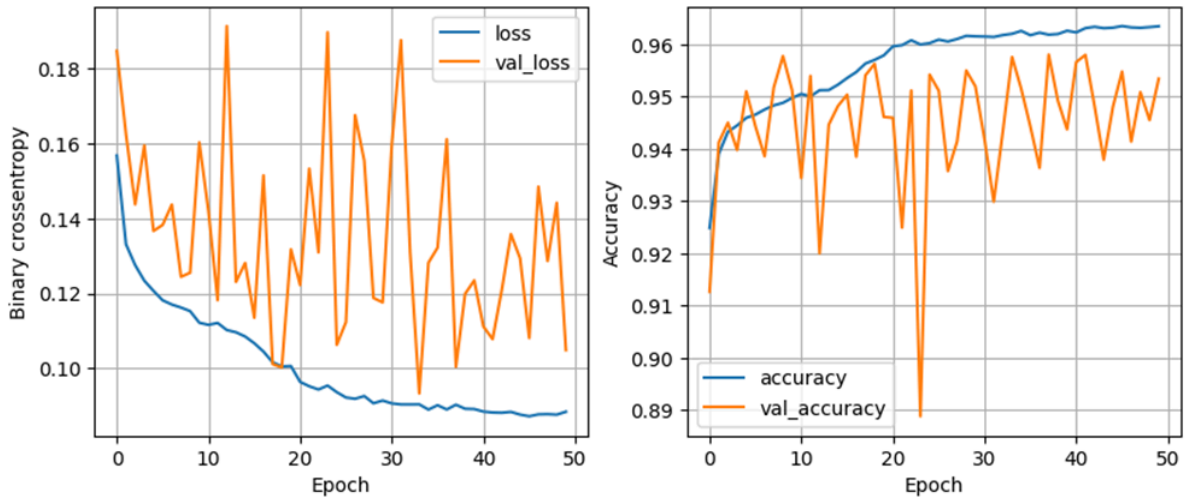
| Metric | k-NN | Logistic Regression | Naive Bayes | SVM | Neural Network |
|---|---|---|---|---|---|
| **Weighted Avg (R)** | 0.93 | 0.90 | 0.57 | 0.93 | 0.97 |
| **Weighted Avg (F1)** | 0.93 | 0.90 | 0.54 | 0.93 | 0.97 |
| **Support (0)** | 18515 | 18515 | 18515 | 18515 | 18515 |
| **Support (1)** | 33020 | 33020 | 33020 | 33020 | 33020 |

The Neural Network model outperformed other models with an accuracy of 0.97. It achieved high precision and recall for both classes, making it the most effective model for this binary classification task.

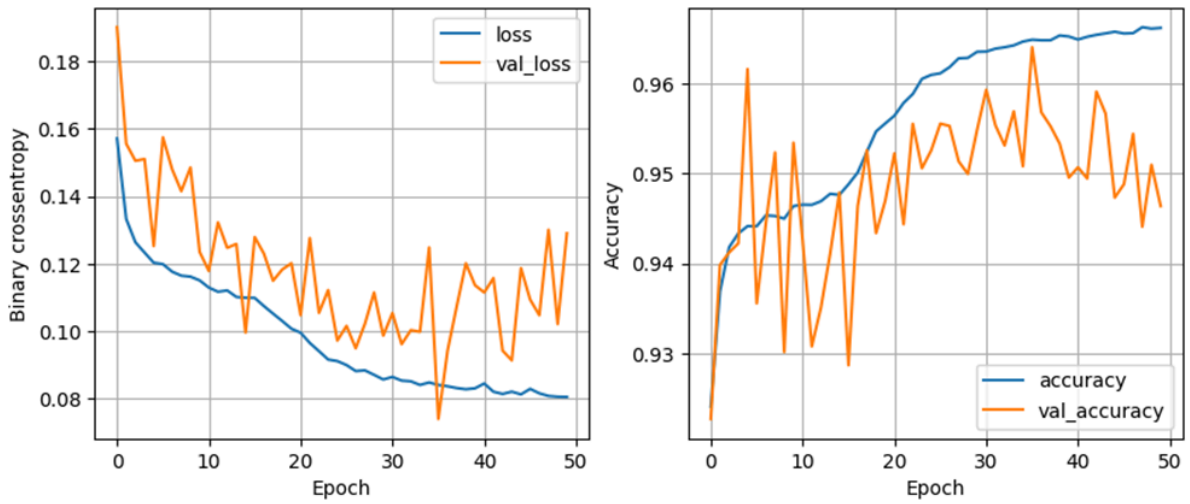
## B. Hyperparameter Tuning of The Neural Network Results

During the hyperparameter tuning of the Neural Network, various configurations were tested to determine the optimal combination of parameters for the best performance. The configurations included variations in the number of nodes, dropout rates, learning rates, and batch sizes. The results of these experiments are illustrated in Figure 1and summarized in Table 3.

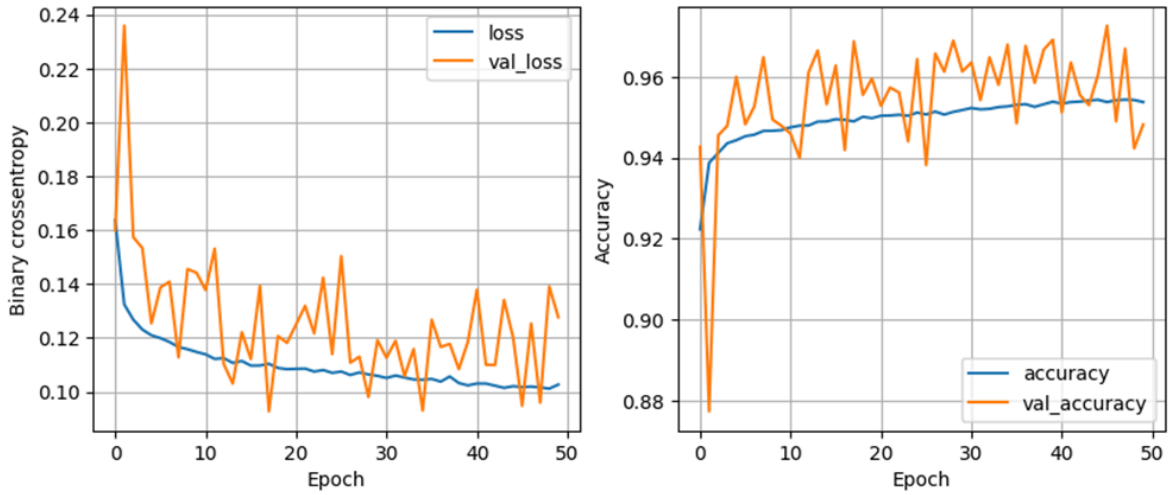
(a)



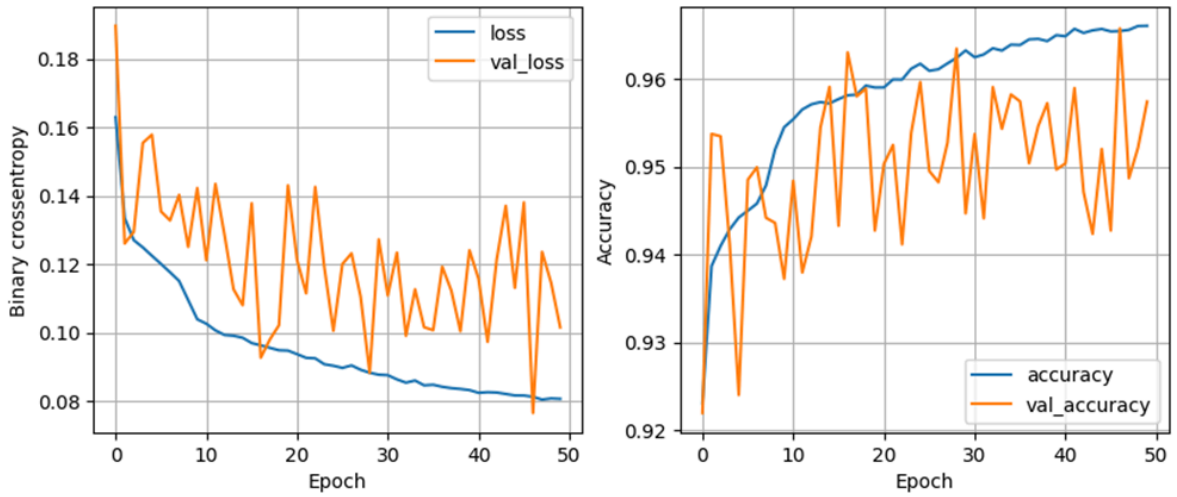

(b)



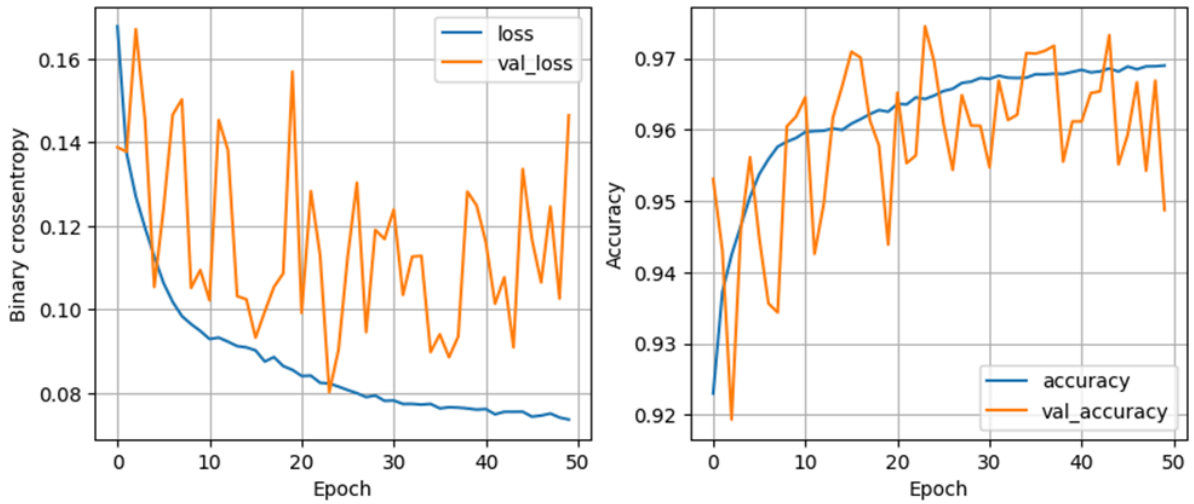
(c)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

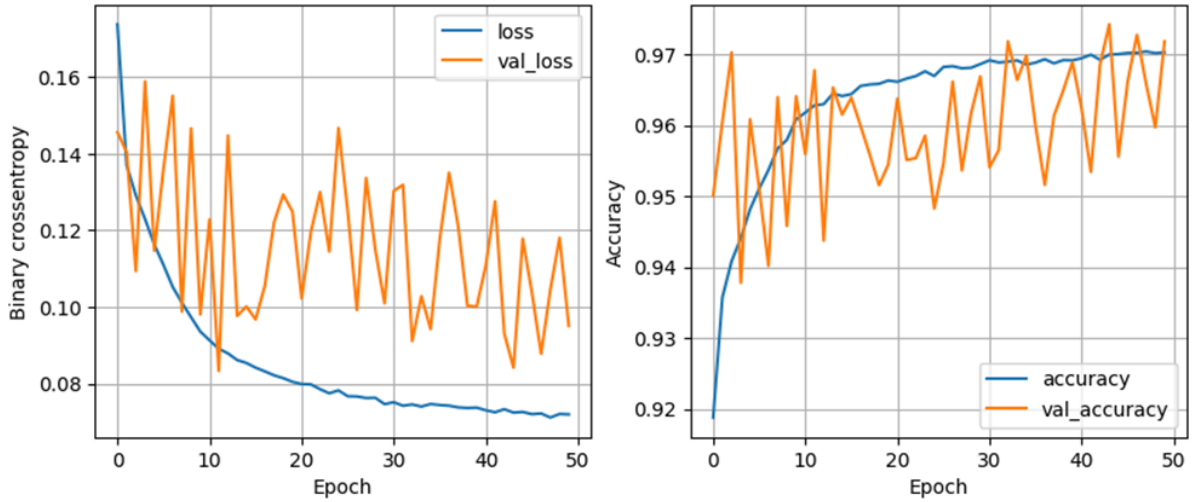
(d)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

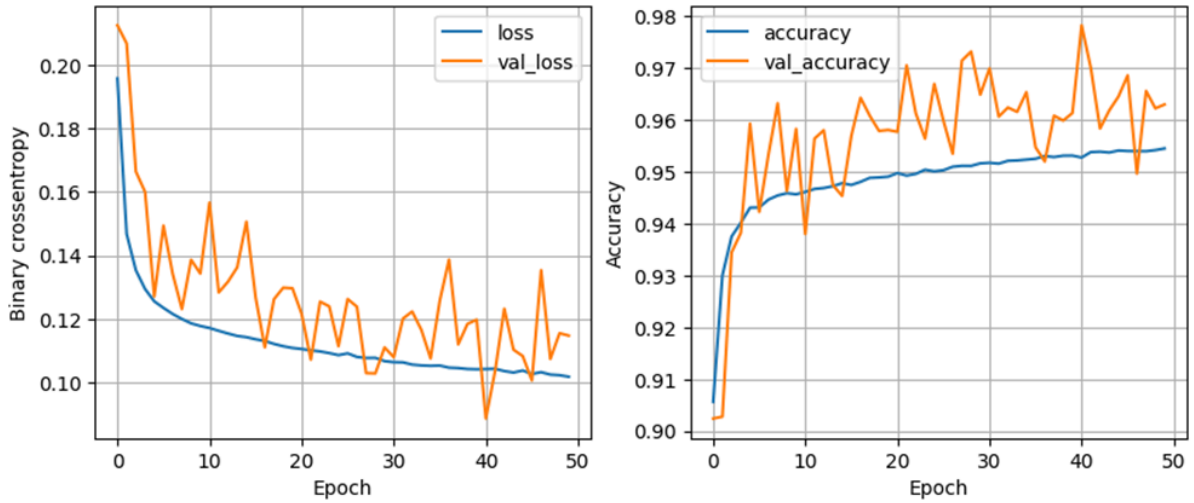
(e)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

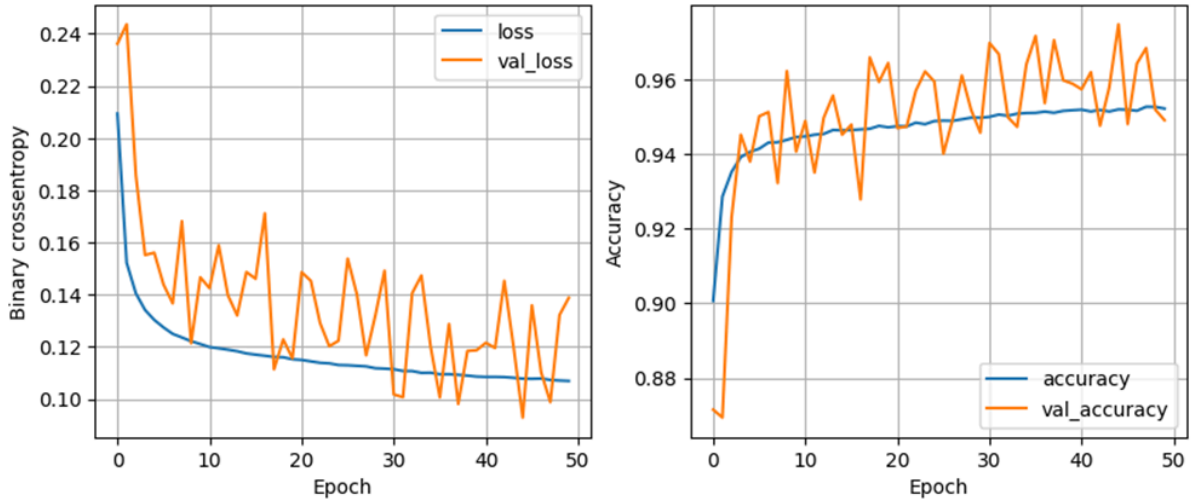
(f)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

(g)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

(h)


loss
val_loss
Binary crossentropy
Epoch
accuracy
val_accuracy
Accuracy
Epoch

(i)

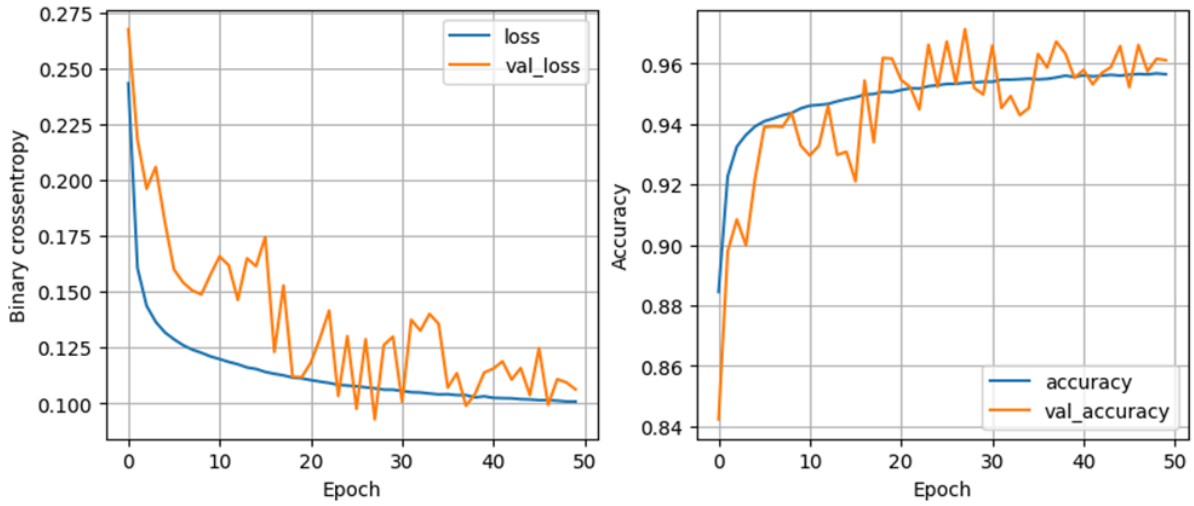



Figure 1: Neural Network Hyperparameter Tuning Results (a) – (i)

Table 3. Hyperparameter Tuning of configuration (a) to (i)

| Configuration | Nodes | Dropout | Learning Rate | Batch Size | Loss | Accuracy |
|---|---|---|---|---|---|---|
| (a) | 16 | 0 | 0.01 | 32 | 0.0911 | 0.9610 |
| (b) | 16 | 0 | 0.01 | 64 | 0.0912 | 0.9622 |
| (c) | 16 | 0 | 0.01 | 128 | 0.1109 | 0.9517 |
| (d) | 16 | 0 | 0.005 | 32 | 0.0875 | 0.9624 |
| (e) | 16 | 0 | 0.005 | 64 | 0.0803 | 0.9650 |
| (f) | 16 | 0 | 0.005 | 128 | 0.0743 | 0.9689 |
| (g) | 16 | 0 | 0.001 | 32 | 0.1056 | 0.9519 |
| (h) | 16 | 0 | 0.001 | 64 | 0.1091 | 0.9518 |
| (i) | 16 | 0 | 0.001 | 128 | 0.1029 | 0.9554 |

Among the tested configurations, configuration (f) achieved the best performance, with a loss of 0.0743 and an accuracy of 0.9689. This configuration utilized 16 nodes, no dropout, a learning rate of 0.005, and a batch size of 128. These results are illustrated in Figure 1, where each subplot corresponds to a specific configuration, displaying the loss and accuracy metrics. This visual representation allows for a clear comparison of the impact of different hyperparameters on the model's performance.

In conclusion, the hyperparameter tuning process identified the optimal configuration for the Neural Network, significantly enhancing its accuracy and reliability in detecting network intrusions.

These results illustrate the varying effectiveness of different machine learning models in classifying network traffic as normal or attack. The Neural Network model showed the highest overall performance, followed by k-NN and SVM, while Naive Bayes lagged behind due to its simplistic approach.

## IV. Conclusion

The experimental results indicate varying degrees of effectiveness across the different machine learning models tested on the UNSW-NB15 dataset for binary classification of network traffic. Here, we delve deeper into the performance of each model, discussing their strengths and weaknesses.

k-Nearest Neighbors (k-NN): The k-NN algorithm demonstrated strong performance with an accuracy of 0.93. It achieved a high precision of 0.97 for the attack class, indicating that when it predicted an attack, it was

correct 97% of the time. The recall for the attack class was 0.92, which means it correctly identified 92% of the actual attacks. However, the precision for the normal class was lower at 0.87, and the recall was higher at 0.96. This suggests that while k-NN is good at identifying normal traffic, it sometimes misclassifies normal traffic as attacks.

Logistic Regression: Logistic Regression also performed well, achieving an accuracy of 0.90. It had a precision of 0.93 for the attack class and 0.85 for the normal class. The recall was 0.91 for the attack class, slightly lower than k-NN, but still indicating good performance in identifying attacks. The lower precision and recall for the normal class suggest that Logistic Regression is slightly less effective at distinguishing normal traffic compared to k-NN.

Naive Bayes: Naive Bayes showed the lowest performance among the models, with an overall accuracy of 0.57. It had a high recall of 0.97 for the normal class but a significantly lower recall of 0.34 for the attack class. This indicates that while Naive Bayes can identify normal traffic well, it struggles significantly with correctly identifying attack traffic, leading to many false positives. The precision for the normal class was 0.45, and for the attack class, it was 0.95, which highlights its limitation in handling the complexity of the dataset due to its simplistic assumption of feature independence.

Support Vector Machine (SVM): SVM performed comparably to k-NN, with an accuracy of 0.93. It achieved high precision (0.97) and recall (0.91) for the attack class, indicating its robustness in identifying attacks. The precision for the normal class was 0.86, and the recall was 0.95, which are slightly lower than k-NN but still reflect strong performance. SVM's effectiveness in high-dimensional spaces and with different kernel functions makes it a solid choice for this task.

Neural Network: The Neural Network model outperformed all other models, achieving the highest accuracy of 0.97. It had very high precision and recall for both the attack (0.98 precision, 0.97 recall) and normal (0.94 precision, 0.96 recall) classes. This indicates that the Neural Network was the most effective in distinguishing between normal and attack traffic, with minimal false positives and false negatives. Its ability to learn complex patterns in the data contributed to its superior performance.

Comparison and Insights: The comparison of these models reveals that while traditional machine learning models like k-NN, Logistic Regression, and SVM perform well, Neural Networks provide a significant edge due to their ability to capture intricate data patterns. Naive Bayes, although simple and efficient, falls short in this context due to its strong independence assumptions which do not hold in the complex network traffic data.

These findings highlight the importance of choosing the right model based on the specific characteristics of the dataset and the classification task. The superior performance of the Neural Network model suggests that deep learning approaches are highly suitable for network intrusion detection tasks, providing high accuracy and reliability in distinguishing between normal and attack traffic.

In conclusion, while traditional models can offer good performance, advanced models like Neural Networks are essential for achieving the highest accuracy in complex and high-dimensional tasks like network intrusion detection. This study underscores the value of evaluating multiple models to find the best fit for a given dataset and problem.

**Strengths of the Study**

- Comprehensive Evaluation: The study evaluated a range of machine learning models, providing a broad perspective on their performance for network intrusion detection.
- Robust Preprocessing: The preprocessing steps ensured that the dataset was clean and appropriately balanced, which is crucial for obtaining reliable results.
- Detailed Analysis: The detailed analysis of each model's performance provides valuable insights into their strengths and weaknesses, aiding in informed decision-making for model selection.

**Limitations and Future Work**

- Dataset Limitations: While the UNSW-NB15 dataset is comprehensive, it is still a synthetic dataset. Real-world traffic data might present additional challenges such as more noise and varying attack patterns.

- Class Imbalance: Despite using oversampling techniques, class imbalance remains a challenge. Future work could explore more sophisticated resampling methods or cost-sensitive learning techniques.
- Model Complexity: The study focused on a limited set of model architectures. Future research could explore more advanced neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), which might capture temporal patterns in network traffic data more effectively.

This study demonstrates that machine learning models, particularly Neural Networks and SVMs, can effectively classify network traffic into normal and attack categories. The findings highlight the potential of these models for enhancing network intrusion detection systems, thereby contributing to the development of more secure and resilient network infrastructures. Future research should focus on addressing the identified limitations and exploring new methodologies to further improve detection accuracy and robustness.

## References

[1] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal*, vol. 25, no. 1–3, 2016, doi: 10.1080/19393555.2015.1125974.

[2] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020, doi: 10.1109/ACCESS.2020.2973730.

[3] J. Vitorino, R. Andrade, I. Praça, O. Sousa, and E. Maia, "A Comparative Analysis of Machine Learning Techniques for IoT Intrusion Detection," 2022, pp. 191–207. doi: 10.1007/978-3-031-08147-7_13.

[4] A. B H, B. S. Akki, H. M. Harshitha, N. R, and V. D.E, "A Survey on Intrusion Detection System using Machine Learning Techniques," *Int J Res Appl Sci Eng Technol*, vol. 11, no. 5, pp. 473–477, May 2023, doi: 10.22214/ijraset.2023.51499.

[5] A. A. Salih and A. M. Abdulazeez, "Evaluation of Classification Algorithms for Intrusion Detection System: A Review," *Journal of Soft Computing and Data Mining*, vol. 02, no. 01, Apr. 2021, doi: 10.30880/jscdm.2021.02.01.004.

[6] R. Zhao, Y. Mu, L. Zou, and X. Wen, "A Hybrid Intrusion Detection System Based on Feature Selection and Weighted Stacking Classifier," *IEEE Access*, vol. 10, pp. 71414–71426, 2022, doi: 10.1109/ACCESS.2022.3186975.

[7] A. Golrang, A. M. Golrang, S. Yildirim Yayilgan, and O. Elezaj, "A Novel Hybrid IDS Based on Modified NSGAII-ANN and Random Forest," *Electronics (Basel)*, vol. 9, no. 4, p. 577, Mar. 2020, doi: 10.3390/electronics9040577.

[8] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, Nov. 2019, doi: 10.1016/j.future.2019.05.041.

[9] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, New York, NY, USA: ACM, Mar. 2006, pp. 16–25. doi: 10.1145/1128817.1128824.

[10] H. Han, H. Kim, and Y. Kim, "An Efficient Hyperparameter Control Method for a Network Intrusion Detection System Based on Proximal Policy Optimization," *Symmetry (Basel)*, vol. 14, no. 1, p. 161, Jan. 2022, doi: 10.3390/sym14010161.

[11] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput Secur*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.