Optimizing Network Traffic Classification Models with a Hybrid Approach for Large-Scale Data

Andrew C Handoko*[1], Hendry[2], Theophilus Wellem[3]
[1,2,3]Faculty of Information Technology, Satya Wacana Christian University
e-mail: *[1]**andrewhandoko925@gmail.com**, [2]hendry@uksw.edu,
[3]theophilus.wellem@uksw.edu

Abstract

The escalating threat of cyberattacks necessitates the development of intrusion detection models that are both accurate and computationally efficient for large-scale network traffic. To address this issue, this study proposes a hybrid approach combining Autoencoder, Convolutional Neural Network (CNN), and XGBoost as an adaptive and lightweight solution for network traffic classification. The key contribution of this research lies in the design of a multi-stage pipeline that performs dimensionality reduction, feature extraction, and final classification. The model was evaluated using the Moore Dataset, which contains complex and high-dimensional network traffic data. The experimental results indicate that the proposed hybrid model achieved a classification accuracy of 99.20% with a testing time of only 0.09 seconds. Furthermore, the pipeline significantly reduced computational load compared to single CNN or XGBoost models. These findings demonstrate that the hybrid approach not only offers high classification performance but also enhances scalability and efficiency, making it suitable for real-world implementation in modern network security systems. Overall, the proposed model presents a promising and practical solution for advancing future intrusion detection systems.

Keywords— Network Traffic Classification, Hybrid, Autoencoder, CNN, XGBoost

1. INTRODUCTION

The escalating threat of cyberattacks is reflected in findings from the National Cyber and Crypto Agency (Badan Siber dan Sandi Negara) in April 2023, which recorded over 27 million network anomalies within a single month, with more than 14 million of them identified as malware [1]. This phenomenon aligns with a 2023 report by the same agency in collaboration with the Indonesia Honeynet Project, which placed Indonesia at the top of the list among ten countries as the primary source of cyberattacks [2]. This increase is not only occurring in Indonesia but also represents a global trend driven by various factors, including international conflicts [3]. In the United States, for example, the number of data breach incidents rose dramatically by 246% between 2014 and 2023 [3]. This surge also translates into significant economic losses, which are projected to continue growing, with estimated global damages from cyberattacks expected to exceed USD 13 trillion within the next four years [3].

In addition to causing global economic losses, the impact of cyberattacks is also strongly felt in Indonesia, particularly in the rapidly growing digital economy and e-commerce sectors, which have not yet been matched by adequate data security measures [4, 5]. Data breaches involving major platforms such as Tokopedia and Bukalapak, along with a fourfold increase in attack incidents in 2020, highlight the high vulnerability of domestic systems to cyber threats [4, 5]. The global surge in software security vulnerabilities has further exacerbated cybersecurity conditions and increased psychological pressure on professionals who are required to respond to threats continuously [3]. The Allianz Risk Barometer 2025 even identifies cyberattacks as the top threat to the business world, underscoring the high level of risk and widespread impact they pose—technically, financially, and socially [3].

As a mitigation measure against the growing risk of cyberattacks, numerous studies have explored and developed machine learning and deep learning methods to enhance the effectiveness of detection systems, particularly in dealing with the increasing complexity of network traffic. For instance, Ayodeji and Melesew (2024) demonstrated that the integration of Software Defined Networking with machine learning algorithms can identify encrypted traffic with up to 99% accuracy while simultaneously improving network service quality [6]. However, deep learning models such as Convolutional Neural Networks (CNNs) still have certain limitations, especially when handling imbalanced datasets. Iwang et al. (2025) reported that although CNNs achieved 99% accuracy on an imbalanced dataset, the low recall and F1-score (35% and 52%, respectively) indicate a tendency of the model to be biased toward the majority class [7]. Moreover, according to Jawad and Sania (2024), deep learning approaches generally rely on labeled data and require substantial computational resources [8]. On the other hand, various algorithms such as K-Nearest Neighbors, Random Forest, and XGBoost have been widely used for intrusion classification. Research by Earum et al. (2022) found that ensemble algorithms such as Random Forest and XGBoost tend to outperform basic models like K-Nearest Neighbors [9]. However, a study by Witcha and Siriporn (2023) revealed that the performance of XGBoost can still be significantly improved through hyperparameter optimization, which can yield F1-scores of up to 99% [10]. Nevertheless, single-algorithm approaches have not yet fully addressed the challenges of classifying large-scale and complex network data.

In response to these limitations, recent research trends indicate a shift toward hybrid approaches and advanced model optimization techniques. Elsedimy and Sara (2025) successfully designed a hybrid model combining Fuzzy C-Means and the Sperm Whale Algorithm, which achieved an accuracy of 96%, outperforming several single algorithms such as ANN, SVM, and Logistic Regression [11]. In addition, Rahmat and Yanif (2023) found that the PCA-MPM model outperformed LSTM-PCA in detecting novel attacks, with accuracy ranging from 90% to 94% [12]. Feature selection techniques have also been shown to significantly enhance model accuracy and efficiency. A study by Sulandri et al. (2021) demonstrated that applying Correlation-Based Feature Selection to the Extreme Learning Machine algorithm improved accuracy from 82% to 98% while also reducing computation time [13]. Further support comes from Ahmad et al. (2024), who showed that the application of ICA and Autoencoder techniques increased the accuracy of MLP from 85% to 94% [14]. Meanwhile, a study by Rehab et al. (2025) emphasized the importance of feature scaling and dimensionality reduction techniques in accelerating computation time, particularly in real-time applications [15].

Building upon the aforementioned findings, this study identifies key gaps in existing methods where CNNs often show bias toward majority classes in imbalanced datasets, XGBoost requires extensive feature sets and careful tuning for efficiency, and standalone dimensionality reduction methods such as PCA or Autoencoders fail to capture complex non-linear traffic patterns [7, 8, 10]. These limitations indicate that single-algorithm approaches remain insufficient for large-scale and complex network data. To address this, the study proposes a hybrid model that integrates Autoencoder, Convolutional Neural Network (CNN), and XGBoost to achieve both high accuracy and computational efficiency in network traffic classification. The Autoencoder will be employed to perform nonlinear dimensionality reduction while preserving essential features, thereby effectively reducing redundancy and enhancing computational efficiency in large-scale intrusion detection systems [16]. CNN is then utilized due to its effectiveness in automatically extracting local patterns from network traffic, similar to how it recognizes features in images, and has been proven effective in real-time anomaly detection systems based on Software Defined Networking [17]. Additionally, approaches such as DeepInsight have demonstrated that non-image data can be transformed into spatial representations and subsequently processed by CNN to extract critical information, yielding promising results [18]. Meanwhile, XGBoost is selected as the final classification algorithm because of its high capability to handle large volumes of data efficiently, its fast training process, and system optimization features that make it highly scalable [19]. The combination of these three algorithms is expected to produce a classification model that is not only robust and adaptive but also optimized for

.

computational time and reduced model complexity in modern network environments.

This hybrid model is designed to optimize the classification process of large-scale network traffic while maintaining performance comparable to using the full feature set, but with shorter computation time and lower model complexity. This approach is highly relevant in addressing big data challenges in network security, including massive traffic volume, the need for real-time detection, and limited computational resources. In addition to enriching academic contributions in the development of intelligent detection systems, this solution also offers practical value for operational implementation, as highlighted by prior studies that emphasize the importance of lightweight and adaptive models for real-time applicability in modern network environments [15, 17].

2. RESEARCH METHOD

The research methodology adopted in this study is designed to evaluate the performance of a hybrid classification model consisting of Autoencoder, CNN, and XGBoost in analyzing network traffic, as illustrated in Figure 1. The process begins with data and environment preparation, followed by data cleaning and preprocessing, then feature reduction using the Autoencoder, feature extraction by the CNN, and final classification using XGBoost. The evaluation is based on several metrics, including accuracy, precision, recall, F1-score, confusion matrix, and computation time. To assess the effectiveness of this approach, comparative experiments are conducted using two single models, namely CNN and XGBoost, and the results are also compared with previous studies to highlight the contribution of the hybrid model.

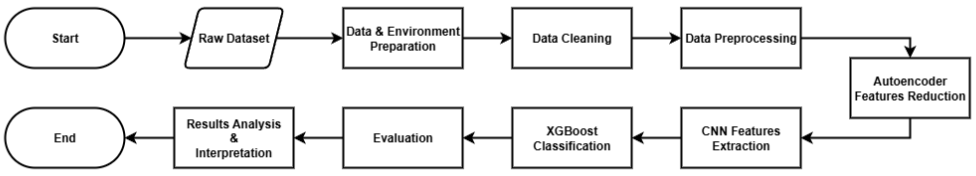

Figure 1. Research Methodology Flowchart

2.1. Proposed Model

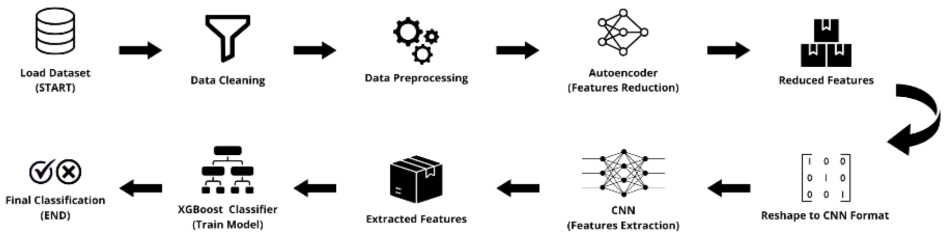
Load Dataset (START)
Data Cleaning
Data Preprocessing
Autoencoder (Features Reduction)
Reduced Features
Reshape to CNN Format
CNN (Features Extraction)
Extracted Features
XGBoost Classifier (Train Model)
Final Classification (END)

Figure 2. Architecture of the Proposed Model

The proposed hybrid model integrates three core components, namely Autoencoder, CNN, and XGBoost to optimize the classification of large-scale network traffic. As illustrated in Figure 2, the architecture follows a multi-stage pipeline that includes dimensionality reduction, feature extraction, and final classification.

In the first stage, an Autoencoder is used to reduce the dimensionality of high-volume network data while retaining essential information. This neural network consists of an encoder that compresses the input into a latent representation and a decoder that attempts to reconstruct the original data, where the reconstruction error indicates the quality of the compression [20, 21, 22, 23]. Unlike linear methods such as PCA, Autoencoders are capable of capturing non-linear patterns [24], which makes them highly effective for processing complex datasets like network traffic. Previous studies have shown that Autoencoders can accelerate training, reduce redundancy, and minimize model complexity in intrusion detection systems [14, 25]. The size and diversity of the Moore dataset further support optimal Autoencoder performance without the typical limitation of insufficient data volume.

After the data has been compacted by the Autoencoder, the CNN component is applied to extract higher-level patterns. While originally designed for image processing, CNNs have been successfully applied to structured non-image data such as network traffic, especially with the help of transformation methods like DeepInsight [26, 27, 28]. A CNN typically includes convolutional layers that detect local patterns, pooling layers that reduce the feature map size while preserving key characteristics, and fully connected layers that generate the output, all enhanced by non-linear activation functions [29]. In this study, the CNN is responsible for learning complex representations from the reduced data, enabling the system to distinguish between different traffic classes more effectively [17, 8]. This automated feature extraction helps reduce manual engineering effort and improves the adaptability of the model in real-world scenarios.

The final stage of the model involves XGBoost, which performs the classification task. XGBoost is a gradient boosting algorithm that builds a sequence of decision trees, where each tree attempts to correct the errors of its predecessor while applying regularization techniques to prevent overfitting [30, 31]. Its efficiency in parallel processing and memory usage makes it well-suited for large-scale data analysis [31, 32]. In addition, XGBoost supports hyperparameter tuning and performs well on multidimensional input features, making it a strong candidate for network traffic classification [10, 15]. In this hybrid architecture, XGBoost receives the features extracted by the CNN and produces the final classification output with high accuracy and speed.

The entire process is visualized in Figure 2, which illustrates the integration of all components in a sequential pipeline. The Autoencoder transforms raw input data into a more compact form, the CNN extracts meaningful patterns from the compressed data, and XGBoost performs efficient and accurate classification. This combination offers a balanced approach that maintains strong classification performance while reducing computational complexity, making it well-suited for intrusion detection systems operating in large-scale and dynamic network environments.

2.2. Dataset

This study utilizes the Moore Dataset, obtained from the University of Cambridge Computer Laboratory, which records 24 hours of network traffic in an active research environment involving thousands of users [33]. The data is divided into ten segments, each approximately 28 minutes in duration, covering between 20,000 and over 66,000 TCP communication flows per segment. After being converted into CSV format, the dataset consists of 377,526 rows and 249 feature columns. Each flow represents a communication session between a client and a server, containing attributes such as connection time, packet count, and transmission duration. The Moore Dataset is known for its rich set of features, including basic statistics (such as packet count, packet size, and inter-arrival time) and TCP protocol information (such as SYN, ACK, and retransmissions) [34].

In addition to its large number of features, another advantage of this dataset lies in the diversity of its network application classes. It contains nine classification labels, namely: Bulk, Database, Interactive, Mail, Services, WWW, P2P, Malicious, Games, and Multimedia [34]. These labels represent various types of network activities commonly found in real-world environments. The Moore Dataset has also proven effective in various statistical approaches such as Naive Bayes due to its ability to reveal traffic patterns from features like packet size and time intervals [35, 36]. With its large scale and wide feature variation, this dataset is highly relevant for testing hybrid models aimed at reducing complexity and computation time in large-scale network traffic classification.

.

2.3. Data Preparation

Table 1. Dataset Summary

Id	4	8	13	14	…	266
1	80	16945	0.000152	0.000683	…	WWW
2	80	16913	0.000161	0.000676	…	WWW
3	80	16917	0.000190	0.000322	…	WWW
4	80	17018	0.000156	0.000269	…	WWW
5	80	16926	0.000170	0.000312	…	WWW

The dataset was obtained from the official Moore Dataset repository in .pcap format, then converted to .csv format using the Weka application. The files were merged into a single file using Python and stored on Google Drive. Subsequently, essential libraries such as NumPy and Pandas were imported, and the dataset was loaded for initial inspection. The original structure of the dataset consists of 377,526 rows and 250 columns, including both numerical and non-numerical features. As shown in Table 1, the first five columns and the last column are presented to provide a general overview. The first column contains an ID that is irrelevant for further processing, while the last column serves as the classification label in string format. All feature columns lack descriptive names and are instead represented by numbers, in accordance with the official documentation [33].

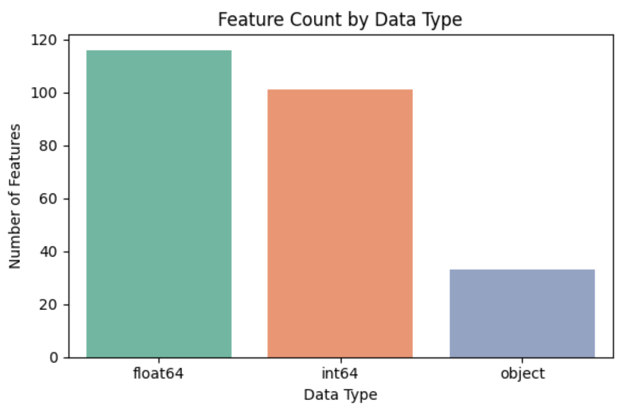

Figure 3. Datatype Distribution

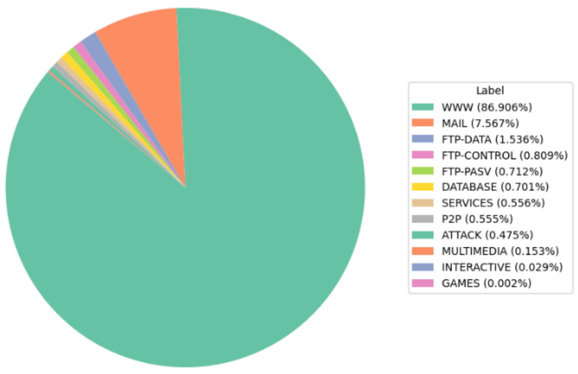

Figure 4. Label Distribution

The initial inspection of the dataset revealed the presence of "?" values that need to be cleaned. Although most of the entries are already in numerical form, there are still 33 features of the object type, including the classification label, as shown in Figure 3. These 33 features must be converted into numerical format to meet the requirements of machine learning algorithms.

Furthermore, Figure 4 illustrates an imbalanced distribution of classification labels, where the "WWW" class is dominant, while "GAMES" and "INTERACTIVE" classes are significantly underrepresented. To address this imbalance and improve model performance, the SMOTE method will be applied in the next preprocessing stage.

2.4. Data Cleaning & Preprocessing

The data cleaning and preprocessing stage began by removing the 'id' column, which is irrelevant for classification, followed by checks for duplicate entries and missing values. Although no duplicates or standard missing values such as NaN were found, invalid entries marked with the character "?" were identified and removed. As a result, 189,027 rows were eliminated, leaving 188,499 valid rows. Due to computational limitations in Google Colab, stratified sampling was performed by selecting a maximum of 2,000 rows per label. Two labels, namely "GAMES" and "MULTIMEDIA," were automatically excluded because their data contained invalid "?" entries.

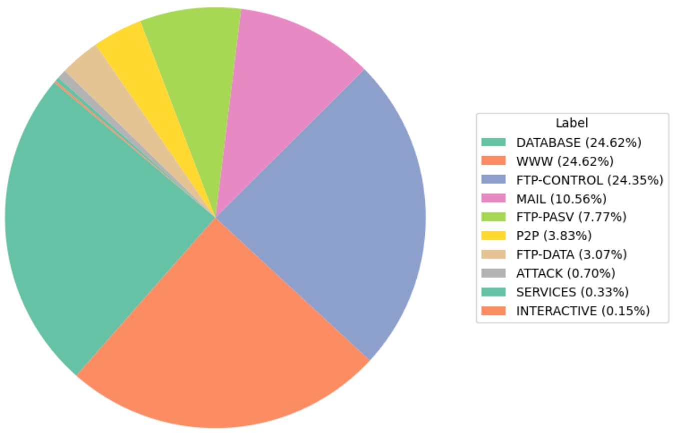

Figure 5. Label Distribution After SMOTE

The dataset was then separated into features (X) and labels (Y), and all columns with object data types were converted to numeric values. The label column (Y) was also encoded into numerical format. Next, the features (X) were normalized using StandardScaler to ensure a consistent data scale. As a result of the balancing process with SMOTE, each label contained an equal number of instances, totaling 181,855 rows per label, making the overall dataset consist of 1,818,550 rows, as illustrated in Figure 5. Ho wever, due to the significant increase in dataset size after oversampling and the computational limitations of the training environment, a further balanced sampling was applied by selecting 2,000 rows per class. This reduction was intended to maintain fairness across classes while ensuring that the model could be trained efficiently within resource constraints. This final dataset was then used for training the model in the subsequent stage.

2.5. Modelling & Evaluation

The modeling phase began with dimensionality reduction using an Autoencoder, which simplified the original 248 features into a more compact yet informative representation. Several variations in the number of features were tested to obtain the optimal result, which was then used as input for the CNN to extract higher-level features. The CNN produced a smaller set of new features that preserved important patterns within the data. All extracted features were stored and used for training the XGBoost classification model, with the data split into 80% for training and 20% for testing. This modeling process was designed systematically and sequentially to ensure that the pipeline operated efficiently and could be easily replicated.

To ensure reproducibility, the parameter settings for each component in the hybrid pipeline were carefully configured based on preliminary testing. The Autoencoder was trained with multiple encoding dimensions, dropout regularization, and the Adam optimizer to obtain compact yet informative features. CNN was applied to extract higher-level representations using convolutional and dense layers with controlled training settings. Finally, XGBoost was optimized with tuned hyperparameters to balance accuracy and efficiency. The complete configuration is summarized in Table 2.

Table 2. Parameter Settings

Component	Parameter Settings
Autoencoder	Encoding dimensions: [100, 110, 120, 130, 140, 150]; Hidden layers: 128–64–(encoding_dim); Activation: ReLU; Dropout: 0.2; Optimizer: Adam (lr=0.0005); Loss: MSE; Batch size: 32; Epochs: 50 with early stopping (patience=5).
CNN	Conv1D: 32 filters, kernel size=3, ReLU; Flatten layer; Dense layer (encoding_dim: 100/75/50); Optimizer: Adam; Loss: MSE; Batch size: 256; Epochs: 50 with early stopping; Validation split: 20%.
XGBoost	n_estimators=100; max_depth=5; learning_rate=0.05; subsample=0.8; colsample_bytree=0.8; reg_alpha=0.1; reg_lambda=1.0; eval_metric=logloss; random_state=42.

Model evaluation was conducted to assess how effectively the model classified different types of network traffic. One of the evaluation metrics used was the confusion matrix, which compares predicted outcomes with actual labels across four categories: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) [37]. TP and TN indicate correct predictions, while FP and FN represent classification errors. In the context of network traffic, FP refers to the incorrect identification of normal traffic as a threat, whereas FN indicates a failure to detect an actual threat. This evaluation is essential for understanding the model's accuracy and the potential risks of misclassification in real-world applications.

From the four categories in the confusion matrix, various evaluation metrics were calculated. Accuracy is the most basic metric, indicating the proportion of correct predictions out of the total data, and is formulated as:

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{1}$$

Accuracy is suitable when the data distribution is balanced, but it can be misleading if one class is significantly more dominant [37]. Therefore, additional metrics such as Precision and Recall are used. Precision indicates how many of the predicted positive instances are actually positive [2], and is defined by the formula:

$$Precision = \frac{TP}{(TP+FP)} \tag{2}$$

In the context of intrusion detection, high precision means that the model rarely triggers false alarms. A simple analogy is that high precision in traffic classification is like a security officer who only stops vehicles that are truly suspicious. On the other hand, recall measures how many of the actual positive cases are successfully identified by the model [30], and is defined by the formula:

$$Recall = \frac{TP}{(TP+FN)} \tag{3}$$

A high recall indicates that the model is very sensitive to detecting attacks, although it may result in a greater number of false alarms. A simple analogy is that high recall in network traffic classification is like a security checkpoint that attempts to stop all potentially dangerous vehicles to ensure none pass through undetected, even if it means occasionally stopping regular vehicles. To achieve a balance between precision and recall, the F1-score is used, which is the harmonic mean of the two [2]:

$$F1 - Score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{4}$$

The F1-score can be analogized as a network security system that strives to remain vigilant without overreacting, by balancing accuracy and alertness in detecting malicious traffic.

In addition to accuracy and classification precision, computational time is also a crucial aspect of evaluation, especially for systems processing large-scale data. Training time measures how long it takes for a model to reach optimal performance, while testing time reflects the model's speed in processing and predicting new data [13]. In practice, both metrics serve as benchmarks for assessing the efficiency and feasibility of a model when deployed in operational environments that require real-time and continuous intrusion detection. All experiments in this study were conducted using a Google Colab environment equipped with a TPU v2–8 and approximately 16 GB of RAM, providing parallelized acceleration suitable for large-scale model training and evaluation.

3. RESULT AND DISCUSSION

Feature reduction was performed on the initial 248 features using an Autoencoder with six variations in the number of features: 100, 110, 120, 130, 140, and 150. Experimental results showed that 110 features yielded the lowest reconstruction loss value of 0.50298, followed by 140 and 120, as presented in Table 3. However, the 110-feature configuration required a relatively longer training time compared to the others. On the other hand, the 130-feature configuration recorded the fastest training time but resulted in the highest reconstruction loss of 1.02790. Among all variations, 120 features offered a balanced trade-off, with a low reconstruction loss value of 0.58333 and the second fastest training time at 138.97 seconds. Considering both reconstruction loss and training time, the 120-feature configuration was selected as the optimal feature representation for further processing with the CNN model.

Table 3. Reconstruction Loss & Training Time Autoencoder

Number of Features	Reconstruction Loss	Training Time (in Seconds)
100	0.59825	185.39
110	0.50298	209.58
120	0.58333	138.97
130	1.02790	110.84
140	0.54394	189.44
150	0.69465	199.17

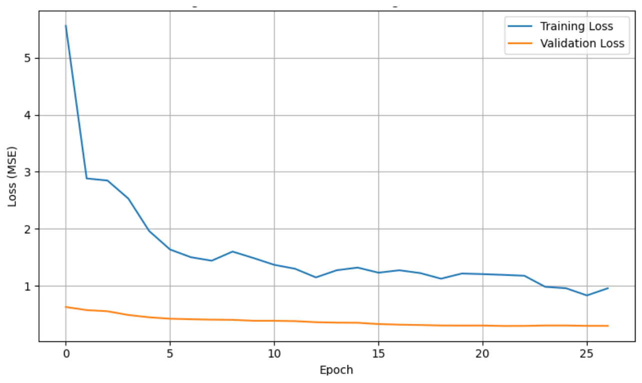

Figure 6. Autoencoder Training and Validation Loss Curve

Figure 6 presents the evaluation results for potential overfitting through an analysis of the training loss and validation loss curves during the training process. The training loss curve shows a sharp decline in the early epochs and tends to stabilize in the subsequent epochs. Meanwhile, the validation loss curve also demonstrates a consistent decrease without any significant spikes or increases. The gap between the two curves remains relatively small throughout the training process, indicating that the model does not experience overfitting. This suggests that the Autoencoder effectively learns patterns from the training data while maintaining stable performance on the validation data. Moreover, it successfully captures essential information from the data without significantly compromising the quality of feature representation. Therefore, the resulting feature reduction is considered adequate and suitable for use in the next processing stage.

Feature extraction was then performed on the 120 features produced by the Autoencoder using a Convolutional Neural Network (CNN). From the training process, a set of 50 features was selected as the final output of the extraction. This selection was based on computational time considerations, as 50 features resulted in the shortest processing time compared to larger feature sets. Additionally, the number of 50 features was deemed sufficiently representative to retain essential information from the reduced network traffic data while also minimizing the complexity of the classification model. Using fewer than 50 features poses a risk of losing critical information, which could negatively affect classification accuracy. To ensure that the CNN feature extraction process functioned effectively, an analysis of the training and validation loss curves was conducted, as shown in Figure 7. The curves exhibit a stable downward trend with no significant gap between the training and validation losses, indicating that the CNN model did not experience overfitting.

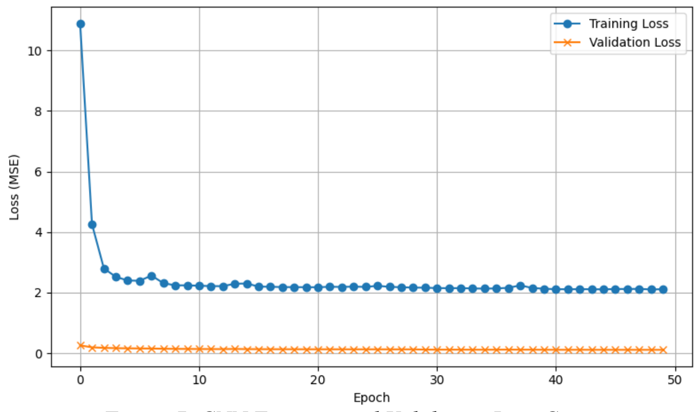

Figure 7. CNN Training and Validation Loss Curve

The features extracted by the CNN were then used as input for the XGBoost classification model. The dataset was split into two parts, with 80% allocated for training and 20% for testing. Evaluation results on the training data demonstrated very high performance, with accuracy, precision, recall, and F1-score all reaching 99.78%. The confusion matrix also indicated that the model was able to correctly classify nearly all data points, showing no signs of overfitting since the performance did not reach a perfect score of 100%, as presented in Table 4.

Table 4. XGBoost Performance

Evaluation Metrics	**On Data Training**	**On Data Testing**
Accuracy	99.78%	99.20%
Precision	99.78%	99.21%
Recall	99.78%	99.20%
F1-Score	99.78%	99.20%
Training Time		9.55 seconds
Testing Time		0.09 seconds

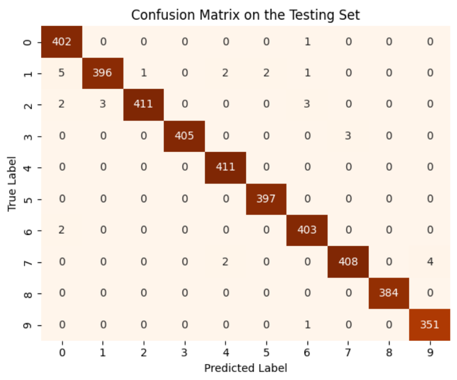

Figure 8. Confussion Matrix

When tested with the testing data, the model maintained consistent performance with an accuracy of 99.20% and comparable precision, recall, and F1-score values. As shown in Figure 8, the confusion matrix for the test data demonstrates a very good classification distribution, with minimal classification errors that are sparsely spread across several classes. The consistency in performance between the training and testing data indicates that the feature reduction and extraction processes successfully simplified the data without degrading the model's performance. It also demonstrates that XGBoost is capable of efficiently classifying network traffic with strong generalization ability.

Table 5. Model Comparison Results

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Training Time**	**Testing Time**
Hybrid (Autoencoder + CNN + XGBoost)	99.20%	99.21%	99.20%	99.20%	138.97 + 107.67 + 9.55 = 256.19 seconds / 4.27 minutes	0.09 seconds
XGBoost	99.98%	99.98%	99.9 %	99.97%	24.7640 seconds	0.11 seconds
CNN	99.92%	99.93%	99.92%	99.93%	464.83 seconds / 7.75 minutes	1.33 seconds

Compared to the CNN and XGBoost models alone, as shown in Table 5, the CNN model required the longest training time of 464.83 seconds, with a testing time of 1.33 seconds. Although its performance was high (99.92%), the computational load was significantly heavier because CNN had to process high-dimensional data without prior dimensionality reduction. On the other hand, the standalone XGBoost model was the fastest and most accurate, with a training time of only 24.76 seconds and performance reaching 99.98%. However, this speed is achievable only under optimal conditions, such as the availability of high computational resources and well-preprocessed data. When handling data with a very large number of features, the efficiency of XGBoost can decline due to the substantially increased computational burden.

Therefore, the hybrid model that combines Autoencoder, CNN, and XGBoost offers a more balanced solution for handling large-scale data with complex features. Unlike Autoencoder and XGBoost, which reduces dimensionality but lacks the ability to capture local feature interactions, or CNN and XGBoost, which can extract patterns but may struggle with very high-dimensional inputs, the three-way combination leverages the strengths of each method. The Autoencoder functions to efficiently reduce the initial dimensionality without losing important information, resulting in more compact features that are easier for the CNN to process. The CNN then extracts relevant local patterns from these features and produces a denser representation. This extracted output is subsequently used as input for XGBoost, which performs the final classification with a significantly lighter computational load because it processes only the features processed by the Autoencoder and CNN rather than the entire original feature set. This is evidenced by the training time of XGBoost within the hybrid pipeline, which is only 9.55 seconds—much shorter compared to the standalone XGBoost model's 24.76 seconds. Although the total training time of the hybrid model is longer, at 256.19 seconds, its testing time is only 0.09 seconds, faster than both the CNN and standalone XGBoost models. With this efficiency, the study results indicate that the hybrid approach not only delivers competitive classification performance but also provides advantages in system stability and scalability, making it more adaptive for real-world scenarios with large data volumes and limited computational resources.

Table 6. Comparison with Previous Studies

Author	Algorithm	Accuracy
[20]	Naive Bayes	20.75%
	Naive Bayes + Kernel Density	37.65%
	Fast Correlation-Based Filter + Naive Bayes	93.38%
	Fast Correlation-Based Filter + Naive Bayes	93.73%
[38]	DNN + K = 10 folds	99.15%
	KNN	98.90%
	SVM	98.56%
[39]	CMSVM (Cost Sensitive Multi-Class SVM)	94%
[40]	SOM-K (Self Organizing Maps KMeans)	90%
This Study	Hybrid (Autoencoder + CNN + XGBoost)	99.20%

As supporting evidence for the performance of the developed approach, this study compares the hybrid model's performance with several methods previously used in related studies on the same dataset, as shown in Table 6. Conventional methods such as Naive Bayes, even when combined with the Fast Correlation-Based Filter, were only able to achieve a maximum accuracy of 93.73%. Other approaches, such as CMSVM and SOM-K, recorded accuracies of 94% and 90%, respectively. Meanwhile, deep learning-based methods, such as the Deep Neural Network with 10-fold cross-validation, achieved an accuracy of 99.15%, slightly below the results of the hybrid model in this study.

The hybrid model combining Autoencoder, CNN, and XGBoost achieved a performance of 99.20%, demonstrating superiority in terms of accuracy compared to other approaches, whether based on single machine learning models or deep learning. In terms of efficiency, the total training time of 256.19 seconds and testing time of 0.09 seconds are considered fast compared to other methods, especially deep learning approaches such as DNN, which typically require longer training times due to cross-validation processes and high network complexity. This indicates that the hybrid approach is not only accurate but also capable of maintaining overall process efficiency.

Thus, the evaluation results demonstrate that the proposed hybrid approach not only excels in performance but is also computationally efficient and capable of handling large-scale data challenges. The combination of Autoencoder, CNN, and XGBoost helps to gradually simplify feature complexity, enabling the final classification process to be faster and lighter. While a more extensive sensitivity analysis, such as varying SMOTE ratios or testing different feature dimensions, was not included within the scope of this study, the experiments already incorporated multiple encoding dimensions for the Autoencoder, which provided an indication of model robustness. Future research could expand this aspect to systematically evaluate the effect of different data balancing strategies and feature variations. Therefore, this model is worthy of consideration as a reliable and scalable solution in the development of artificial intelligence-based network traffic detection systems.

4. CONCLUSION

This study proposed a hybrid classification model for network traffic analysis by integrating Autoencoder, CNN, and XGBoost. The model was designed to reduce feature dimensionality, extract higher-level representations, and perform efficient classification on large-scale data. Experimental results showed that the hybrid model achieved an accuracy of 99.20%, with a total training time of approximately 256.19 seconds and a testing time of 0.09 seconds, indicating that the approach delivers competitive classification performance while maintaining fast inference.

As shown in Table 5, standalone XGBoost attained the highest accuracy of 99.98%, while CNN also achieved a competitive result but with significantly higher training and testing times. The hybrid pipeline, although slightly lower in accuracy, provides more balanced advantages by controlling feature complexity, reducing input redundancy, and achieving the fastest inference (0.09 seconds) compared to CNN (1.33 seconds) and XGBoost (0.11 seconds). When compared with previous studies on the Moore dataset, the proposed model remains competitive and demonstrates adaptability for large-scale environments. These findings suggest that the hybrid approach constitutes a reliable and scalable solution for network traffic classification and can serve as a foundation for further research and development in more dynamic settings such as IoT ecosystems or cloud-based infrastructures.

REFERENCES

[1] A. Gofur, R. F. Aji, and H. Kurniawan, "Pengukuran kesadaran keamanan informasi pegawai: Studi kasus PT Meshindo Jayatama," J. Tek. Inf. dan Ilmu Komput., vol. 11, no. 2, pp. 315–320, Apr. 2024, doi: 10.25126/jtiik.20241128106.

[2] K. Inayah and K. Ramli, "Analisis kinerja intrusion detection system berbasis algoritma random forest menggunakan dataset unbalanced honeynet BSSN," J. Tek. Inf. dan Ilmu Komput., vol. 4, no. 11, pp. 867–876, Aug. 2024, doi: 10.25126/jtiik1148911.

[3] F. Hering, O. Hinz, J. Pfeiffer, and W. v. d. Aalst, "The Damocles sword of cyber attacks," Bus. Inf. Syst. Eng., vol. 67, pp. 141–147, Feb. 2025, doi: 10.1007/s12599-025-00933-7.

[4] R. Sham, W. Christian, D. Daud, and M. H. Miraz, "Modelling a sustainable digital economy among e-commerce users in Indonesia," in Proc. 13th AMER Int. Conf. Quality of Life (AicQoL2025), Pangkor Island, Malaysia, 2025. [Online]. Available: www.e-iph.co.uk (accessed Jun. 15, 2025).

[5] F. Fachri, "Optimasi keamanan web server terhadap serangan brute-force menggunakan penetration testing," J. Tek. Inf. dan Ilmu Komput., vol. 10, no. 1, pp. 51–58, Feb. 2023, doi: 10.25126/jtiik.2023105872.

[6] A. O. Salau and M. M. Beyene, "Software defined networking based network traffic classification using machine learning techniques," Sci. Rep., vol. 14, no. 20060, Aug. 2024, doi: 10.1038/s41598-024-70983-6.

[7] I. M. A. Surya, T. A. Cahyanto, and L. A. Muharom, "Deep learning dengan teknik early stopping untuk mendeteksi malware pada perangkat IoT," J. Tek. Inf. dan Ilmu Komput., vol. 12, no. 1, pp. 21–30, Feb. 2025, doi: 10.25126/jtiik.2025128267.

[8] J. H. Kalwar and S. Bhatti, "Deep learning approaches for network traffic classification in the Internet of Things (IoT): A survey," arXiv preprint, arXiv:2402.00920. [Online]. Available: https://arxiv.org/abs/2402.00920 (accessed May 27, 2025).

[9] E. Mushtaq, F. Shahid, and A. Zameer, "A comparative study of machine learning models for malware detection," in Proc. 2022 19th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST), Islamabad, Pakistan, Aug. 16–20, 2022, pp. 677–681, doi: 10.1109/IBCAST54850.2022.999054.

[10] W. Chimphlee and S. Chimphlee, "Hyperparameters optimization XGBoost for network intrusion detection using CSE-CICIDS 2018 dataset," IAES Int. J. Artif. Intell., vol. 13, no. 1, pp. 817–826, 2024, doi: 10.11591/ijai.v13.i1.pp817826.

[11] E. L. Elsedimy and S. M. M. Abohashish, “An intelligent hybrid approach combining fuzzy C-means and the sperm whale algorithm for cyber attack detection in IoT networks,” Sci. Rep., vol. 15, no. 1005, 2025, doi: 10.1038/s41598-024-79230-4.

[12] R. Budiarto and Y. D. Kuntjoro, “Analisis perilaku pendeteksian entitas serangan untuk internal menggunakan kombinasi model prediksi memori dan metode PCA,” J. Tek. Inf. dan Ilmu Komput., vol. 10, no. 6, pp. 1223–1232, 2023, doi: 10.25126/jtiik.2023107123.

[13] Sulandri, A. Basuki, and F. A. Bachtiar, “Metode deteksi intrusi menggunakan algoritma extreme learning machine dengan correlation-based feature selection,” J. Tek. Inf. dan Ilmu Komput., vol. 8, no. 1, pp. 103–110, 2021, doi: 10.25126/jtiik.202183358.

[14] A. Sanmorino, S. Suryati, R. Gustriansyah, S. Puspasari, and N. Ariati, “Feature extraction vs fine-tuning for cyber intrusion detection model,” J. Infotel, vol. 16, no. 2, pp. 302–315, 2024, doi: 10.20895/infotel.v16i2.996.

[15] R. H. Serag, M. S. Abdalzaher, H. A. A. Elsayed, and M. Sobh, “Software defined network traffic classification for QoS optimization using machine learning,” J. Netw. Syst. Manag., vol. 33, no. 41, 2025, doi: 10.1007/s10922-025-09911-6.

[16] A. Kumar, R. Radhakrishnan, M. Sumithra, P. Kaliyaperumal, B. Balusamy, and F. Benedetto, “A scalable hybrid autoencoder–extreme learning machine framework for adaptive intrusion detection in high-dimensional networks,” Future Internet, vol. 17, no. 5, p. 221, 2025, doi: 10.3390/fi17050221.

[17] H. Liu and H. Wang, “Real-time anomaly detection of network traffic based on CNN,” Symmetry, vol. 15, no. 6, p. 1205, 2023, doi: 10.3390/sym15061205.

[18] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, “DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” Sci. Rep., vol. 9, p. 11399, Aug. 2019, doi: 10.1038/s41598-019-47765-6.

[19] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” arXiv preprint, arXiv:1603.02754v3, Jun. 10, 2016. [Online]. Available: https://arxiv.org/abs/1603.02754 (accessed Jun. 30, 2025).

[20] J. Cui, L. Bai, G. Li, Z. Lin, and P. Zeng, “Semi-2DCAE: A semi-supervision 2D CNN autoencoder model for feature representation and classification of encrypted traffic,” PeerJ Comput. Sci., vol. 9, p. e1635, Nov. 2023, doi: 10.7717/peerj-cs.1635.

[21] Y. Song, S. Hyun, and Y.-G. Cheong, “Analysis of autoencoders for network intrusion detection,” Sensors, vol. 21, p. 4294, 2021, doi: 10.3390/s21134294.

[22] W. Wenxu, F. S. Sabrina, J. Jang-Jaccard, A. Singh, and Y. Wei, “Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset,” IEEE Access, vol. 9, pp. 140136–140146, 2021, doi: 10.1109/ACCESS.2021.3116612.

[23] B. Sae-ang, W. Kumwilaisak, and P. Kaewtrakulpong, “Semi-supervised learning for defect segmentation with autoencoder auxiliary module,” Sensors, vol. 22, p. 2915, 2022, doi: 10.3390/s22082915.

[24] F. S. Nugraha and H. F. Pardede, “Autoencoder untuk sistem prediksi berat lahir bayi,” J. Tek. Inf. dan Ilmu Komput., vol. 9, no. 2, pp. 235–244, Apr. 2022, doi: 10.25126/jtiik.202293868.

[25] I. Ramadhanti, A. Prasetiadi, and I. K. A. Iqsyahiro, “Clothing recommendation and face swap model based on VGG16, autoencoder, and facial landmark points,” J. Tek. Inform., vol. 5, no. 1, pp. 19–29, Feb. 2024, doi: 10.52436/1.jutif.2024.5.1.1016.

[26] Yuliska, D. H. Qudsi, J. H. Lubis, K. U. Syaliman, and N. F. Najwa, “Analisis sentimen pada data saran mahasiswa terhadap kinerja departemen di perguruan tinggi

menggunakan convolutional neural network," J. Tek. Inf. dan Ilmu Komput., vol. 8, no. 5, pp. 1067–1076, Oct. 2021, doi: 10.25126/jtiik.202184842.

[27] F. Y. Santoso, E. Sediyono, and H. D. Purnomo, "Optimalisasi hyper parameter convolutional neural networks menggunakan ant colony optimization," J. Tek. Inf. dan Ilmu Komput., vol. 11, no. 2, pp. 243–248, Apr. 2024.

[28] C. M. Bachri and W. Gunawan, "Deteksi email spam menggunakan algoritma convolutional neural network (CNN)," J. Eduk. Penelit. Inform., vol. 10, no. 1, pp. 88–94, Apr. 2024.

[29] B. Nugroho and E. Y. Puspaningrum, "Kinerja metode CNN untuk klasifikasi pneumonia dengan variasi ukuran citra input," J. Tek. Inf. dan Ilmu Komput., vol. 8, no. 3, pp. 533–538, Jun. 2021, doi: 10.25126/jtiik.202184515.

[30] Z. S. Dhahir, "A hybrid approach for efficient DDoS detection in network traffic using CBLOF-based feature engineering and XGBoost," J. Future Artif. Intell. Technol., vol. 1, no. 2, pp. 174–190, Sep. 2024, doi: 10.62411/faith.2024-33.

[31] D. Kurnia, M. I. Mazdadi, D. Kartini, R. A. Nugroho, and F. Abadi, "Seleksi fitur dengan particle swarm optimization pada klasifikasi penyakit Parkinson menggunakan XGBoost," J. Tek. Inf. dan Ilmu Komput., vol. 10, no. 5, pp. 1083–1094, Oct. 2023, doi: 10.25126/jtiik.2023107252.

[32] D. Niu, J. Zhang, L. Wang, K. Yan, T. Fu, and X. Chen, "A network traffic anomaly detection method based on CNN and XGBoost," in Proc. 2020 Chinese Autom. Congr. (CAC), 2020, pp. 1–6, doi: 10.1109/CAC51589.2020.9327030.

[33] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Tech. Rep. RR-05-13, Dept. Comput. Sci., Queen Mary Univ. London, Aug. 2005.

[34] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in Lect. Notes Comput. Sci., Mar. 2005.

[35] A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in Proc. SIGMETRICS'05, Banff, AB, Canada, Jun. 2005, pp. 50–60.

[36] D. Zuev and A. W. Moore, "Traffic classification using a statistical approach," in Proc. 6th Passive and Active Meas. Workshop (PAM), Boston, MA, Mar./Apr. 2005.

[37] R. Gustriansyah, N. Suhandi, S. Puspasari, and A. Sanmorino, "Machine learning method to predict the toddlers' nutritional status," J. Infotel, vol. 16, no. 1, pp. 32–43, Feb. 2024, doi: 10.20895/infotel.v15i4.988.

[38] M. B. Umair, Z. Iqbal, M. Bilal, J. Nebhen, T. A. Almohamad, and R. M. Mehmood, "An efficient internet traffic classification system using deep learning for IoT," Comput. Mater. Contin., vol. 71, no. 1, pp. 407–421, 2022, doi: 10.32604/cmc.2022.020727.

[39] S. Dong, "Multi-class SVM algorithm with active learning for network traffic classification," Expert Syst. Appl., vol. 176, p. 114885, 2021, doi: 10.1016/j.eswa.2021.114885.

[40] S. Zhao, Y. Xiao, Y. Ning, Y. Zhou, and D. Zhang, "An optimized K-means clustering for improving accuracy in traffic classification," Wirel. Pers. Commun., 2021, doi: 10.1007/s11277-021-08435-x.